NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

From Prediction to Profit: Evaluating S&P 500 Forecasting Models Using Machine Learning

Sofoklis N. Strompolas

Supervisor: **Panagiotis Stamatopoulos,** Assistant Professor

ATHENS

JULY 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Από την Πρόβλεψη στο Κέρδος: Αξιολόγηση Μοντέλων Πρόβλεψης S&P 500 με τη Χρήση Μηχανικής Μάθησης

Σοφοκλής Ν. Στρόμπολας

Επιβλέπων: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2023

BSc THESIS

From Prediction to Profit: Evaluating S&P 500 Forecasting Models Using Machine Learning

Sofoklis N. Strompolas

S.N.: 1115201700153

SUPERVISOR: **Panagiotis Stamatopoulos,** Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Από την Πρόβλεψη στο Κέρδος: Αξιολόγηση Μοντέλων Πρόβλεψης S&P 500 με τη Χρήση Μηχανικής Μάθησης

Σοφοκλής Ν. Στρόμπολας

Α.Μ.: 1115201700153

ΕΠΙΒΛΕΠΩΝ: Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

ABSTRACT

This thesis presents a comprehensive exploration into the prediction of the S&P 500 closing prices using a multitude of prediction models and trading strategies. Capitalizing on the advancements in Financial Technology (FinTech), it employs Statistical (ARIMA), Machine Learning (Support Vector Regression, SVR), and Deep Learning (Long Short-Term Memory, LSTM) methodologies to forecast the S&P 500 closing prices. These models are applied to four types of data, each differentiated by forecasting horizons and sets of features.

The study starts with the acquisition and preprocessing of financial and economic data, followed by feature selection, after which various models are built and trained. The predictive performance of these models is then evaluated traditionally and also tested in different trading simulation algorithms. This offers a dual lens to assess the models' capabilities, both from a predictive accuracy perspective and a trading performance viewpoint.

The research offers insights into the application of technology in financial prediction and trading, and aims to contribute to the existing body of knowledge in the areas of predictive modeling. Moreover, it serves as an educative journey for anyone seeking to explore the world of stock market predictions and trading strategy formulation.

The results of this comparative analysis show that each of the models has its unique strengths in specific market scenarios. However, overall, the LSTM model consistently outperformed others in predicting the S&P 500's closing prices, as well as proving its efficiency in the simulated trading strategies, affirming the potential of deep learning approaches in financial forecasting. The SVR, while demonstrating good forecasting errors, its performance in trading simulations was relatively inferior. Meanwhile, the traditional ARIMA model, although ranking last in most of the comparisons, made a commendable effort for a statistical approach, highlighting its continued relevance in financial prediction.

The findings from this research are hoped to inspire further exploration and improvements in the combination of finance and technology.

SUBJECT AREA: Financial Technology (FinTech)

KEYWORDS: S&P 500, Financial Prediction, Statistical Modeling (ARIMA), Machine Learning (Support Vector Regression, SVR), Deep Learning (Long Short-Term Memory, LSTM), Stock Market Forecasting, Trading Simulation, Trading Strategies, Financial Technology (FinTech), Financial Data Science, Predictive Modeling, Algorithmic Trading

ΠΕΡΙΛΗΨΗ

Αυτή η πτυχιακή εργασία παρουσιάζει μια ολοκληρωμένη διερεύνηση στην πρόβλεψη των τιμών κλεισίματος του S&P 500 χρησιμοποιώντας μια πληθώρα μοντέλων πρόβλεψης και στρατηγικών συναλλαγών. Αξιοποιώντας τις εξελίξεις στη Χρηματοοικονομική Τεχνολογία (FinTech), χρησιμοποιεί μεθοδολογίες Στατιστικές Μοντελοποίησης (ARIMA), Μηχανικής Μάθησης (Support Vector Regression, SVR) και Βαθιάς Μάθησης (Long Short-Term Memory, LSTM) για να προβλέψει τις τιμές κλεισίματος του S&P 500. Αυτά τα μοντέλα εφαρμόζονται σε τέσσερις τύπους δεδομένων, καθένας από τους οποίους διαφοροποιείται ανάλογα με τους ορίζοντες πρόβλεψης και τα σύνολα χαρακτηριστικών.

Η έρευνα ξεκινά με την απόκτηση και προεπεξεργασία χρηματοοικονομικών και οικονομικών δεδομένων, ακολουθούμενη από επιλογή χαρακτηριστικών, μετά την οποία κατασκευάζονται και εκπαιδεύονται διάφορα μοντέλα. Η προγνωστική απόδοση αυτών των μοντέλων στη συνέχεια αξιολογείται παραδοσιακά, και δοκιμάζεται επίσης σε διαφορετικούς αλγορίθμους επενδυτικών στρατηγικών για προσομοίωση συναλλαγών. Αυτό προσφέρει διπλή αξιολόγηση των δυνατοτήτων των μοντέλων, τόσο από την άποψη της προγνωστικής ακρίβειας όσο και από την άποψη της απόδοσης συναλλαγών.

Η έρευνα προσφέρει πληροφορίες για την εφαρμογή της τεχνολογίας στις χρηματοοικονομικές προβλέψεις και συναλλαγές, και στοχεύει να συμβάλει στο υπάρχον σύνολο γνώσεων στους τομείς της προγνωστικής μοντελοποίησης. Επιπλέον, χρησιμεύει ως ένα εκπαιδευτικό ταξίδι για όποιον θέλει να εξερευνήσει τον κόσμο των προβλέψεων χρηματιστηρίου και της διαμόρφωσης στρατηγικής συναλλαγών.

Τα αποτελέσματα αυτής της συγκριτικής ανάλυσης δείχνουν ότι κάθε ένα από τα μοντέλα έχει τα μοναδικά πλεονεκτήματά του σε συγκεκριμένα σενάρια αγοράς. Ωστόσο, συνολικά, το μοντέλο LSTM ξεπέρασε σταθερά τα άλλα στην πρόβλεψη των τιμών κλεισίματος του S&P 500, καθώς και στην απόδειξη της αποτελεσματικότητάς του στις προσομοιωμένες στρατηγικές συναλλαγών, επιβεβαιώνοντας τις δυνατότητες των προσεγγίσεων βαθιάς μάθησης στις χρηματοοικονομικές προβλέψεις. Το SVR, ενώ παρουσίαζε καλά λάθη πρόβλεψης, η απόδοσή του στις προσομοιώσεις συναλλαγών ήταν σχετικά κατώτερη. Εν τω μεταξύ, το παραδοσιακό μοντέλο ARIMA, αν και κατατάσσεται τελευταίο στις περισσότερες συγκρίσεις, έκανε μια αξιέπαινη προσπάθεια για μια στατιστική προσέγγιση, τονίζοντας τη συνεχιζόμενη συνάφειά του στις χρηματοοικονομικές προβλέψεις.

Τα ευρήματα αυτής της έρευνας αποσκοπούν να εμπνεύσουν περαιτέρω εξερεύνηση και βελτιώσεις στο συνδυασμό των οικονομικών και της τεχνολογίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Χρηματοοικονομική Τεχνολογία (FinTech)

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: S&P 500, Χρηματοοικονομική Πρόβλεψη, Στατιστική Μοντελοποίηση ARIMA, Μηχανική Μάθηση SVR, Βαθιά Μάθηση LSTM, Πρόβλεψη Χρηματιστηρίου, Επενδυτική Προσομοίωση, Επενδυτικές Στρατηγικές, Χρηματοοικονομική Τεχνολογία FinTech, Χρηματοοικονομική Επιστήμη Δεδομένων, Προβλεπτική Μοντελοποίηση, Αλγοριθμική Συναλλαγή

Dedicated to all who played a part...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Rationale and context

The inception of this research is rooted in a personal interest in the FinTech industry, a domain that merges the latest advancements in technology with traditional financial services. This field fuses the worlds of finance and technology, revolutionizing the financial industry and how we manage and interact with money [31].The potential of this sector sparked a curiosity, particularly to understand one of its most complex components, the stock market.

This interest led to the exploration of the stock market's components, and eventually to the complex task of stock price prediction. Despite the inherent volatility and complex nature of financial markets [79], and even a certain level of mistrust [33], the solution may lie in technology. The rapid development of artificial intelligence has demonstrated a promising avenue for addressing this problem.

In particular, machine learning and deep learning techniques have shown potential in unveiling patterns in these seemingly chaotic market behaviors [77]. Therefore, this thesis aims to harness these technologies to evaluate their effectiveness in predicting stock market movements.

1.2 Purpose and aim of the study

This study aims to explore the multidimensional world of stock market prediction. The primary concern is comparing different predictive methods and architectures, such as statistical models, machine learning techniques, and deep learning methodologies. The study engages in a comparison of stock indices, features, and prediction horizons, scrutinizing performance and efficacy across these dimensions.

An essential aspect of this research is to evaluate the impact of different features on the predictive models' performance. Although the efficient-market hypothesis, which will be explained in future chapters, suggests that stock market prices already reflect all available information, many researchers dispute this conclusion. Therefore, many different extrinsic sources of data are used for stock market prediction [41]. This study compares the results obtained from models that use solely market data against those that incorporate additional information, aiming to assess whether incorporating non-market features can enhance the accuracy of stock market predictions.

Beyond the methods, the research investigates the performance of these models across different stock indices. This comprehensive analysis provides an opportunity to appreciate the versatility and robustness of prediction methods in varied market conditions and sectors.

Furthermore, this research aims to investigate the temporal aspect of stock market predictions. It will compare the performance of these models over different forecast horizons, both short and long-term. This comparison offers valuable insights into the differing complexities and challenges posed by various prediction horizons, providing insights into the unique challenges and trade-offs that different prediction horizons may pose.

In essence, this study embodies a purposeful journey aimed at enriching the understand-

ing of stock market prediction strategies. It seeks to offer valuable insights in the finance industry and provide a pathway for future investigations in this domain.

1.3 Structure of the thesis

This thesis is thoughtfully structured into six main chapters, each serving a distinct purpose in the overall narrative of the research. It serves to progress the reader through a journey starting from the broad concept of the prediction of the stock market and gradually moving towards the precise methodologies and results. Each chapter's main purpose and the progression of the research narrative are outlined in the next page in Table 1.1.

Chapter 1, named 'Introduction' presents the rationale behind the research, introduces the purpose and aims of the study, and provides a broad overview of the thesis structure. In Chapter 2, or the 'Background', readers are introduced to the basics of artificial intelligence, the complexities of the financial market, and the principles of stock predictions. This chapter provides necessary context, establishing a solid foundation for the upcoming sections and ultimately linking these sections to stock market prediction. This is followed by 'Literature Review', outlined as Chapter 3, that offers a critical examination of the current academic landscape. This chapter reviews existing prediction methodologies, evaluates their effectiveness, and identifies gaps in the present body of knowledge. Chapter 4, referred to as 'Approach', details the research design. This chapter describes the research plan, procedures for data selection and gathering, the process of model selection and creation, training and steps for validation. Here, the execution of the study takes place where the models are created and evaluated using advanced metrics, trading strategies. In Chapter 5, titled 'Results', the study's findings are presented. This section delves into the analysis and interpretation of the results, along with suggestions for potential performance improvements. The final chapter, Chapter 6 or 'Conclusion', provides a comprehensive summary of the key findings and their implications on the field of financial market predictions. It concludes with a thoughtful reflection on potential future research directions.

Table 1.1: Structure of the Thesis

Chapter	Description
1. Introduction	Provides the rationale behind the research, introduces the purpose and aims of the study, and provides a broad overview of the thesis structure.
2. Background	Introduces the basics of artificial intelligence, the complexities of the financial market, and the principles of stock predictions.
3. Literature Review	Reviews existing prediction methodologies, evaluates their effectiveness, and identifies gaps in the present body of knowledge.
4. Approach	Describes the research plan, procedures for data selection and gathering, the process of model selection and creation, training, steps for validation, and evaluation.
5. Results	Presents the study's findings, analyzes and interprets the results, and provides suggestions for potential performance improvements.
6. Conclusion	Provides a comprehensive summary of the key findings and their implications on the field of financial market predictions. Concludes with a thoughtful reflection on potential future research directions.

2. BACKGROUND

This chapter is designed to provide the reader with the fundamental knowledge necessary to understand the rest of the research work. This chapter will lay the groundwork by introducing the principles of financial markets, stock predictions, and the basics of artificial intelligence. This section essentially forms the backbone of the thesis, linking these various concepts together in the context of stock market prediction.

2.1 Understanding artificial intelligence

Artificial Intelligence (AI) is a broad field of study that focuses on creating machines and software that mimic human intelligence. The goal of AI is to develop systems capable of performing tasks that normally require human intellect, such as understanding natural language, recognizing patterns, solving problems, and making decisions [76].

2.1.1 The basics of machine learning

Machine Learning (ML) is a subfield of artificial intelligence that aims to teach computers the ability to learn from and make decisions or predictions based on data. It seeks to construct algorithms and statistical models that machines, specifically computers, can use to perform specific tasks without explicit instruction, instead relying on patterns and inference derived from data [58].

2.1.1.1 Definition and types of machine learning

Machine Learning, involves computer programs that automatically improve their performance through experience [58]. The types of ML, as seen in the Figure 2.1 below, are mainly categorized into supervised learning, unsupervised learning, and reinforcement learning.

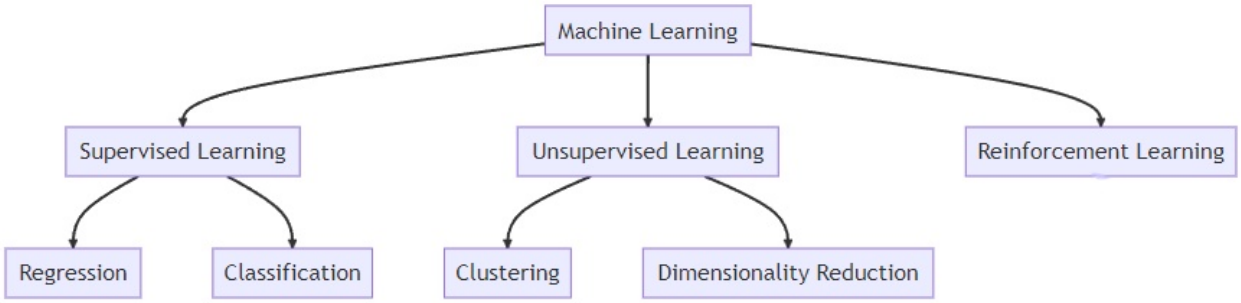

Figure 2.1: Machine Learning Types

2.1.1.2 Supervised learning

Supervised learning, one of the most common types of machine learning, involves training a model to predict outputs from a given set of inputs using a labeled dataset. Each example

in the labeled dataset consists of an input vector and a corresponding desired output value. During the training phase, the model uses this dataset to learn a function that maps the inputs to the correct outputs. Once trained, the model should be able to use this function to make accurate predictions for new, previously unseen instances based on the knowledge it has acquired during training. Supervised learning tasks primarily fall into two categories: regression and classification [58].

- **Regression:** In regression tasks, the goal is to predict a continuous output value. The model is trained with a dataset where the true output values are known. It learns to understand the relationship between the input features and the output value, and applies this understanding to predict the output for new instances.

  A widely-used regression algorithm is linear regression. Linear regression assumes a linear relationship between the input features and the output value and learns a linear function that minimizes the difference between the predicted and true output values. However, in real-world problems where the relationship may not be strictly linear, more complex regression models like polynomial regression or decision trees for regression might be used [58] [87].

- **Classification:** In classification tasks, the output is discrete rather than continuous. This involves predicting which category or class a new instance belongs to based on its features. In multi-class classification problems, there are more than two classes to predict.

  There are several algorithms used for classification, including logistic regression, decision trees, and support vector machines. Logistic regression, despite its name, is used for binary classification tasks. It learns a logistic function that models the probability that a given input point belongs to a specific class. Decision trees build a tree-like model of decisions and their possible consequences, while support vector machines construct hyperplanes in a high-dimensional space to separate different classes [58].

2.1.1.3 Unsupervised learning

Unsupervised learning involves the training of machine learning models on a dataset without labeled outputs. The model needs to find structure in the data and learn the underlying patterns or distributions on its own. This makes unsupervised learning particularly useful for exploratory analysis or when little is known about the relationships in the data [61]. There are primarily two types of unsupervised learning task among others: clustering and dimensionality reduction.

- **Clustering:** Clustering is a task that groups data instances into subsets, known as clusters, based on their similarity. The aim is to make instances in the same cluster more similar to each other than to those in other clusters. This is useful in a variety of applications, such as customer segmentation, image segmentation, and anomaly detection.

  Various algorithms can be used to perform clustering, including K-means, hierarchical clustering, and DBSCAN. K-means is one of the most widely used clustering algorithms. It groups data into K distinct, non-overlapping clusters based on their distances from K centroids, which are determined iteratively [54].

- **Dimensionality Reduction:** Dimensionality reduction is a technique that reduces the number of input variables in a dataset while retaining the essential information. It can be helpful in visualizing high-dimensional data, improving the efficiency of other machine learning algorithms, and mitigating issues related to the curse of dimensionality (problems that arise when dealing with high-dimensional data).

  Principal Component Analysis (PCA) is a widely-used linear dimensionality reduction technique. PCA transforms the original variables to a new set of variables, the principal components, which are orthogonal (uncorrelated), and account for the largest possible variance in the data [43].

  There are also non-linear dimensionality reduction methods like t-SNE and UMAP that are especially useful when the underlying structure of the data is not linear.

  While unsupervised learning can uncover hidden patterns and simplify complex data, its outcomes can sometimes be hard to interpret, and its results depend highly on the quality and structure of the input data [36].

2.1.1.4 Reinforcement learning

Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with its environment. The agent performs certain actions in an environment to achieve a goal, receives feedback in terms of rewards or penalties, and adjusts its actions accordingly. The objective of the agent is to learn a policy, a strategy that dictates the action the agent should choose given the current environment state [89].

2.1.1.5 Evaluating machine learning models

Evaluating machine learning models is a crucial aspect of any machine learning project. It allows us to understand the performance of the model, and whether the model is achieving its desired objectives. The choice of evaluation metrics depends on the specific task and the business requirements. Here, we will discuss some common metrics used in supervised learning tasks - for both regression and classification. These metrics provide quantitative ways to measure how well a model's predictions align with actual data [77].

- **Regression Metrics:** In the case of regression tasks, the model's predicted values ($\hat{y}$) are compared to the actual known values (y) from the data. Additionally, the model aims to predict a continuous value, and we often use the following metrics [97][77][52] [62] [99]:
  - **Mean Absolute Error (MAE):** This is the average of absolute differences between the predicted and actual values. It gives an idea of how wrong the predictions were.
  - **Mean Squared Error (MSE):** This is similar to MAE, but squares the difference before summing them all instead of using the absolute value. This means that MSE is more sensitive to outliers compared to MAE.
  - **Root Mean Squared Error (RMSE):** This is the square root of the MSE. The RMSE is expressed in the same units as the output, which can make it more interpretable than the MSE.

  - **Mean Absolute Percentage Error (MAPE)** This is the average of the absolute percent difference between predictions and actual observations.
  - **R-squared (Coefficient of Determination):** This measures the proportion of the variance in the dependent variable that can be predicted from the independent variable(s). It provides a measure of how well future outcomes are likely to be predicted by the model.

- **Classification Metrics:** For classification tasks, the concepts of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) are essential [77]. These arise from the concept of a confusion matrix, a table layout that visualizes the performance of a classification algorithm. The confusion matrix for binary classification is a 2x2 table:

Table 2.1: Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

The four terms represent:

  - True Positives (TP): The cases in which the model predicted positive, and the truth is also positive.
  - False Positives (FP): The cases in which the model predicted positive, but the truth is negative.
  - True Negatives (TN): The cases in which the model predicted negative, and the truth is also negative.
  - False Negatives (FN): The cases in which the model predicted negative, but the truth is positive.

These terms form the foundation for many evaluation metrics for classification tasks such as Precision, Recall, F1 Score, and others. In classification tasks, where the model predicts a discrete label, common metrics include [72] [77] [29]:

  - **Accuracy:** This is the ratio of the correctly predicted observations to the total observations. While it's straightforward, it can be misleading if the classes are imbalanced.
  - **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positives.
  - **Recall:** Recall (Sensitivity) is the ratio of correctly predicted positive observations to the all observations in actual class. These are especially useful when the costs of false positives and false negatives are very different.
  - **F1 Score:** This is the harmonic mean of Precision and Recall, and tries to find the balance between precision and recall.
  - **Area Under ROC (Receiver Operating Characteristic) Curve (AUC-ROC):** ROC Curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity. The area under this curve is a measure of separability, or how much the model is capable of distinguishing

between classes.The Area Under the ROC Curve (AUC-ROC) is a numerical measure of a model's discrimination capability. It ranges from 0 to 1, with 1 denoting a perfect classifier and 0.5 signifying no better than random guessing. AUC-ROC is a robust measure, considering the model's performance across all thresholds, making it effective especially when dealing with imbalanced classes or differing misclassification costs.

Table 2.2 summarizes the evaluation metrics mentioned above, that are used for regression and classification. This table provides a condensed view of the metrics, offering a convenient reference for understanding and comparing the different evaluation standards applied in machine learning tasks [77].

2.1.2 Machine learning in stock market prediction

Artificial intelligence (AI), especially machine learning (ML), has sparked a substantial amount of interest and investment from financial institutions. These organizations prioritize the application of a variety of machine learning (ML), deep learning (DL), and reinforcement learning (RL) methods, including supervised and unsupervised ML, natural language processing (NLP), and data science, to enhance their investment strategies, gather new insights into the competitive landscape, and ultimately boost profits and outpace rivals [77].

2.1.2.1 ”Black box”

The primary challenge with many ML systems, particularly deep neural networks, is that they function as ”black boxes”. This is due to the difficulty of comprehending their inner workings post-training. Basic models like Linear Regression (Regression) and Decision Trees (Classification) have interpretable structures and fewer parameters that do not require further explanatory techniques. In contrast, complex models like Deep Neural Networks, with millions of parameters (weights), are often referred to as ”black boxes” as their behavior remains indecipherable even when their structure and weights are known [77].

ML enables computers to analyze massive amounts of information in search of trends and patterns, which is essential in the business of trading. Machine learning algorithms are skilled at digesting large volumes of data to uncover patterns difficult for humans to discern [77].

2.1.2.2 Machine learning in quantitative finance

Quantitative finance, which involves the use of mathematical and statistical methods to analyze financial data, increasingly employs machine learning techniques to create more accurate forecasts and enhance the performance of financial models. Here are some examples of machine learning applications in quantitative finance [77]:

- **Algorithmic trading:** ML algorithms create trading strategies that can automatically assess massive volumes of financial data and execute trades based on the algorithm's predictions.

Table 2.2: Summary of Evaluation Metrics

Evaluation Metric	Description	Formula
MAE (Mean Absolute Error)	Average of absolute differences between predictions and actual observations	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
MSE (Mean Squared Error)	Average of the squares of the differences between predictions and actual observations	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
RMSE (Root Mean Squared Error)	Square root of the average of the squared differences between predictions and actual observations	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
MAPE (Mean Absolute Percentage Error)	Average of the absolute percent difference between predictions and actual observations	$\frac{100\%}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$
R-squared (Coefficient of Determination)	Proportion of the variance in the dependent variable that is predictable from the independent variable(s)	$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$
Accuracy	Ratio of correctly predicted observations to the total observations	$\frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$
Precision	Ratio of correctly predicted positive observations to the total predicted positive observations	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$
Recall (Sensitivity)	Ratio of correctly predicted positive observations to all observations in actual class	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$
F1 Score	Harmonic Mean of Precision and Recall	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
AUC-ROC (Area Under the Receiver Operating Characteristic)	A single value summary of the ROC curve representing the model's ability to distinguish between classes across all thresholds.	Calculated as the two-dimensional area under the entire ROC curve.

- **Risk management:** ML can reveal risky trends in large financial datasets, aiding financial firms in risk management.
- **Portfolio optimization:** ML can identify the best investments for a portfolio using factors like volatility, risk, and return.
- **Forecasting:** ML enables the analysis of past financial data to predict future market patterns, exchange rates, among other parameters.

- **Sentiment analysis:** ML can analyze social media and news to determine public sentiment about a company, product, or industry.

Traditional financial models can be integrated with machine learning models to enhance predictions and outcomes. However, not all ML models are appropriate for all problems. The model selection depends on the problem and the available data, so collaboration between machine learning and finance professionals is crucial [77].

2.1.2.3 Machine learning methods in stock market prediction

ML's potential to analyze vast amounts of data and detect complex patterns has led to its extensive application in financial forecasting. Various types of machine learning models are employed in stock market prediction, each with unique advantages, strengths, and application areas [50]:

- **Linear Regression (LR)**: One of the most straightforward ML algorithms, Linear Regression is often used to predict a continuous value like the future price of a stock. LR models the relationship between two variables by fitting a linear equation to observed data, allowing us to understand the impact of changes in independent variables on the dependent variable.
- **Logistic Regression (LR)**: Used primarily for binary classification problems, it's useful in predicting whether a stock price will increase (1) or decrease (0). Despite its name, Logistic Regression is a classification algorithm that uses the logistic function to model a binary dependent variable.
- **Decision Tree (DT)**: These are interpretable models often used for both classification and regression tasks. For example, they could help determine if a stock would rise or fall based on a set of variables like the company's earnings, the sector's performance, and macroeconomic indicators.
- **Random Forest (RF)**: An ensemble learning method that operates by constructing multiple decision trees during training and outputting the majority vote of individual trees for classification problems or average for regression problems.
- **Support Vector Machine (SVM)**: SVMs are primarily used for classification but can be adapted for regression. In stock market prediction, an SVM might categorize stocks into ”likely to rise” and ”likely to fall”.
- **k-Nearest Neighbors (kNN)**: A simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). kNN has been used in statistical estimation and pattern recognition in the stock market as early as the 1970s.
- **Deep Learning (DL)**: DL algorithms, such as artificial neural networks with numerous layers, have been a revelation in the field of stock market prediction. They can model complex non-linear relationships and are particularly effective when working with large amounts of data.

2.1.2.4 Feature selection

Feature selection is an essential step in building an effective ML model for stock market prediction. Features can include historical prices, company financials, market indicators, or even alternative data like social media sentiment or news data [50]. The quality and relevance of selected features directly impact the model's prediction accuracy.

2.1.2.5 Regression and classification

Stock prices are a type of time-series data, and predicting future prices is a common time-series problem. Depending on the nature of the problem, ML in stock market prediction may involve regression or classification [50]:

- **Regression:** Regression analysis is a statistical method for predicting a dependent variable based on the values of at least one independent variable. In the context of stock market prediction, the dependent variable could be the future price of a stock, while the independent variables could be historical prices, volumes, etc.
- **Classification:** Classification models, on the other hand, are used to predict discrete labels, such as whether a stock price will go up or down. The classification models need to be trained on a labeled dataset.

Machine Learning has revolutionized stock market prediction by offering sophisticated algorithms capable of analyzing complex patterns and relationships in data. While traditional econometric models are still valid and widely used, the ability of ML models to work with large-scale, high-dimensional data and to model non-linear relationships gives them an edge in many forecasting scenarios. However, building a successful ML model for stock market prediction also requires careful data pre-processing, feature selection, and model validation, as well as a deep understanding of the financial markets themselves [50].

2.1.3 The basics of deep learning

Deep learning is a subset of machine learning techniques that is characterized by the use of networks of simple concepts, which are arranged in various architectures. This arrangement allows computers to learn complex concepts from simple nodes that are graphically interconnected through multiple layers [32]. The resurgence of deep learning was primarily driven by probabilistic or Bayesian models, such as Deep Belief Networks (DBN). These networks are composed of nodes that represent random variables, each having probabilistic relationships with one another [32] [38].

In recent years, a different type of model, known as Artificial Neural Networks (ANN), has gained increasing popularity. These networks consist of nodes that represent neurons, which are generated during the training process. The rise in popularity of ANNs signifies a shift in the deep learning landscape, highlighting the continuous evolution and adaptability of these techniques [64].

2.1.3.1 History

The history of deep learning is a fascinating journey that traces back to the 1940s and 1950s, with the development of the earliest artificial neural networks. However, it wasn't until the 1980s and 1990s that the first algorithms capable of training deep architectures were developed. Despite these early advancements, deep learning didn't gain significant attention until the 2000s, when advancements in computational power and the availability of large datasets made it possible to train complex models. The field has since seen rapid growth and development, with deep learning models now being used in a wide range of applications, from image recognition to natural language processing.

2.1.3.2 Differences with machine learning

Despite being a subset of machine learning, deep learning differs from its parent field in several key ways. Deep learning constructs algorithms in various layers to create an artificial neural network capable of learning and making intelligent decisions independently. In contrast, machine learning requires algorithms to parse data, learn from it, and then make informed decisions. Deep learning requires a large amount of data and high-performance hardware, while machine learning can work with smaller datasets and less powerful hardware. In deep learning, the model itself can create new features, whereas in machine learning, features must be accurately and precisely recognized by the users. Deep learning solves problems on an end-to-end basis, while machine learning decomposes larger tasks into smaller ones and combines the results. Despite the longer training time, deep learning models often achieve a higher accuracy rate than machine learning models. Furthermore, deep learning eliminates the challenging and complex feature engineering phase present in machine learning. However, it's worth noting that deep networks require high-end graphical processing units, which can be expensive and time-consuming to train with large datasets [19].

2.1.3.3 Neural networks

Artificial Neural Networks (ANNs), considered to be the heart of deep learning algorithms, are computational models designed based on the structure of the human brain and are aimed at solving complex problems by emulating human intelligence [21]. A typical neural network architecture, as depicted in Figure 2.2, consists of an input layer that receives data, one or more hidden layers that process and transform the information, and an output layer that produces the final prediction or output. The Figure 2.2 includes the following components:

- Input Layer (Blue): This is where the neural network receives input from the data set. It's represented by "Input Neurons".
- Hidden Layers (Green): These are the layers in between the input and output layers where the actual processing is done via a system of weighted connections. The diagram includes two hidden layers, "Hidden Layer 1" and "Hidden Layer 2", for illustration purposes.
- Output Layer (Red): The final layer, "Output Neurons", is where the neural network makes a decision about the input data based on the learning from the hidden layers.

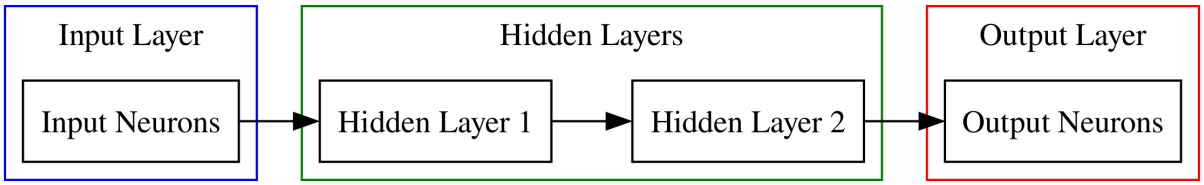

Figure 2.2: Neural Network Structure

2.1.3.4 Artificial neuron

ANNs are composed of simple, interconnected processing units called artificial neurons or nodes, which perform calculations on incoming data. ANNs store and process information distributed across multiple nodes, rather than reserving a single unit for any particular pattern.

The diagram Figure 2.3, showing the structure of an artificial neuron, includes the following components:

- Inputs (Blue): These are the input values for the neuron. In this case, there are three inputs labeled ”Input 1”, ”Input 2”, and ”Input 3”.
- Weights (Green): These are the weights assigned to each input. The weights are used to amplify or dampen the input signals.
- Bias (Red): This is an additional input that allows the activation function to be shifted to the left or right, to better fit the data.
- Activation Function (Orange): This function processes the inputs, weights, and bias to produce an output. It decides whether a neuron should be activated or not.
- Output (Purple): This is the final output of the neuron after processing the inputs, weights, bias, and activation function.

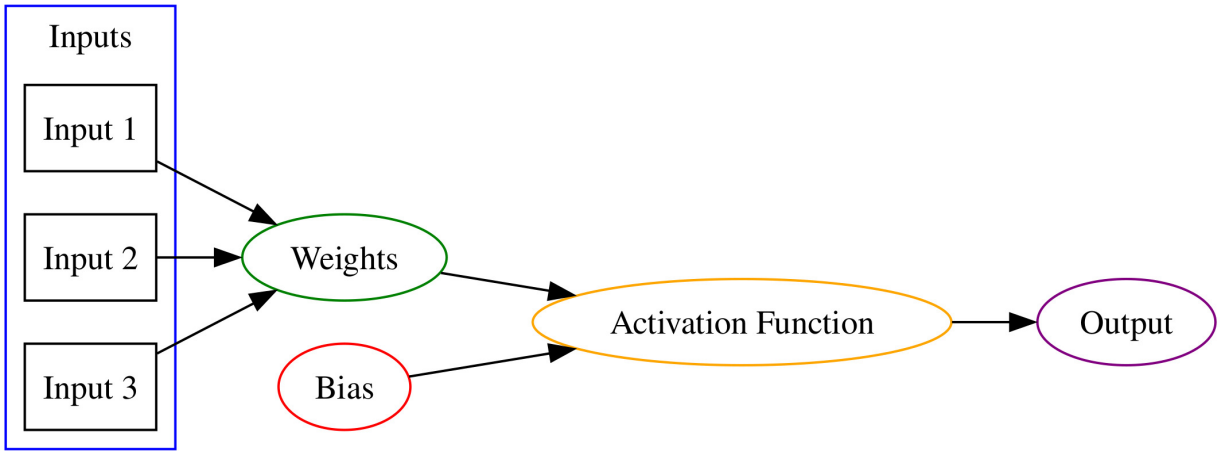

Figure 2.3: Artificial Neuron

Each artificial neuron, a simplified abstraction of a biological neuron, performs a two-step computation. Firstly, each neuron multiplies the incoming data (inputs) by respective synaptic weights, and then aggregates them, adding the neuron’s bias value [64]. This can be mathematically represented as:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{2.1}$$

where:

z is the aggregate sum. w_i are the weights. x_i are the inputs. b is the bias. The sum is over n inputs.

- **Activation Function:** The next step involves passing the aggregated value, z, through an activation function, also referred to as the squashing function. The activation function, denoted as g(z), restricts the output of the neuron to a specific range, generating the final output, also known as activation (a):

  The output or activation of the neuron, denoted as a, is given by passing z through the activation function $g(z)$:

  $$a = g(z) \tag{2.2}$$

  Here, $g(z)$ is the activation function.

  Activation functions play a crucial role in ANNs, and several types are used depending upon the architecture and the specific problem at hand. Some of the most commonly used activation functions are the Rectified Linear Unit (ReLU), sigmoid, and hyperbolic tangent (tanh) [32]. The ReLU activation function is popular in Feed-Forward Neural Networks (FFNNs) due to its computational efficiency and because it does not suffer from the vanishing gradient problem, a common issue in deep neural networks with sigmoid activation functions. Sigmoid and tanh functions are commonly used in recurrent networks for their ability to maintain values within a specified range [32][64].

  ReLU, sigmoid, and tanh are some commonly used activation functions. Their representation is as follows:

  ReLU (Rectified Linear Unit):

  $$g(z) = \mathsf{max}(0, z) \tag{2.3}$$

  Sigmoid:

  $$g(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

  Hyperbolic Tangent (tanh):

  $$g(z) = \mathsf{tanh}(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.5}$$

- **Layers:** A layer in a deep learning model is a collection of neurons that process a set of input data. Each layer is like a filter or transformation that the data passes through. The layers closer to the input data are usually responsible for learning simple patterns, and as the data progresses through the network, the patterns become more complex.

- **Learning:** Learning in the context of ANNs involves the adaptation of the network's parameters, namely weights and biases, based on the input data. Learning techniques can be categorized based on how weights are adjusted in the network (supervised, unsupervised, reinforcement learning) and how data is made available to the network (offline/batch or online learning) [21].

  The learning process begins with initializing weights and biases randomly. The network then iteratively updates these parameters to reduce the error rate, which is calculated by comparing the network's prediction with the true values using a loss function [64]. The error is then propagated back to the network in a process called backpropagation, which adjusts the weights using an optimization method like Stochastic Gradient Descent [74]. This process is repeated over multiple iterations or epochs until a defined number of iterations is achieved, or the error rate falls below a satisfactory threshold.

2.1.3.5 Training

- **Backpropagation:** Backpropagation is a fundamental algorithm in training neural networks. It is the method by which we adjust the weights of the network based on the error at the output. During the forward pass, the input data is passed through the network to generate a prediction. The prediction is compared with the actual output, and the error is calculated. Backpropagation then calculates the gradient of this error with respect to the network's weights, and these gradients are used to update the weights, ideally reducing the error. This forms the basis of what is known as Gradient Descent, an optimization algorithm [74].

- **Gradient Descent:** Gradient descent is an optimization algorithm used to minimize a function iteratively. In the context of machine learning and deep learning, the function we typically want to minimize is the cost or loss function, which measures how well our model is performing given the current parameter values [32].

  The concept behind gradient descent begins with the initial values for our model's parameters and then update these values iteratively to move towards the point of minimum loss. At each iteration, we compute the gradient (i.e., the vector of partial derivatives) of the cost function at the current point, and then we take a step in the direction opposite to the gradient (as the gradient points in the direction of steepest ascent, but we want to descend) [32].

  The mathematical representation of the gradient descent algorithm is as follows:

  1. Initialize the weights (parameters) randomly. Let's denote these weights as θ.
  2. Calculate the gradient of the cost function $J(\theta)$ with respect to each parameter θ_i. This gives us a gradient vector. The gradient of the cost function at a particular point is given by:

$$\nabla J(\theta) = \left[\frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, ..., \frac{\partial J}{\partial \theta_n}\right]$$

  3. Update each weight according to the rule:

$$\theta_i := \theta_i - \alpha \frac{\partial J}{\partial \theta_i}$$

where α is the learning rate (a hyperparameter that determines the step size during each iteration), and $\frac{\partial J}{\partial \theta_i}$ is the gradient of the cost function with respect to θ_i.

4. Repeat steps 2 and 3 until convergence, i.e., until the difference in cost between successive iterations is smaller than a pre-defined threshold, or until a pre-set number of iterations have been performed.

Gradient descent comes in different flavors, such as batch gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent, each of which uses a different number of training examples to compute the gradient at each step. In deep learning, mini-batch gradient descent and its variants (like Adam and RMSprop) are most commonly used due to their balance between computational efficiency and convergence speed [32].

- **Overfitting and Underfitting:** A major challenge in training deep learning models is striking the right balance to avoid overfitting and underfitting. Overfitting occurs when the model learns the training data too well, essentially memorizing noise and outliers, leading to poor performance on unseen data. Underfitting, conversely, is when the model fails to capture the underlying patterns of the data, performing poorly on both training and test data.

  Solutions to these issues are inherently different. For overfitting, techniques such as dropout and L1/L2 regularization are effective as they limit the complexity of the model. For underfitting, solutions often involve increasing the model's complexity, such as adding more layers or nodes to the neural network, or reducing bias in the model [32].

- **Regularization:** Regularization is a technique used to prevent overfitting, a situation where the model performs exceptionally well on the training data but poorly on the test or unseen data. Essentially, regularization adds a penalty term to the loss function, effectively limiting the magnitude of the model parameters and therefore the model complexity. The two main types of regularization in deep learning are L1 and L2 regularization. L1 regularization tends to produce sparse weights, while L2 regularization encourages smaller weights but does not necessarily produce sparse solutions.

2.1.3.6 Supervised learning architectures

Supervised learning architectures in deep learning are built to learn from labeled training data, where the models predict output values based on input data and learn through a continuous comparison between these predictions and actual results.

- **Feed-forward Neural Networks (FFNN):** FFNNs are widely used in deep learning architectures. An FFNN consists of an input layer representing the input example, one or more hidden layers, and an output layer. The output of each layer serves as the input for the next layer. During training, activations are propagated forward across the network, and the error rate is propagated backward from the output layer to the input layer through a process known as backpropagation [64].

- **Recurrent Neural Networks (RNN):** RNNs maintain a representation of previously seen input data, making them ideal for handling sequential data. RNNs contain

loops that allow one or more passes of the same input, with the network maintaining a state representation of each pass. This allows for variable length inputs and outputs. However, typical RNNs struggle to retain information over long periods due to the ”vanishing gradient” problem, where gradients of the loss function become exponentially small as they are propagated back through the layers of the network, resulting in slower learning or failure to learn for earlier layers. To overcome this, variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks have been developed [32].

LSTM networks are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber in 1997 to solve the vanishing gradient problem that affects the training of traditional RNNs [37].

LSTMs are designed to remember information for prolonged periods. This capability is achieved through a complex cell state, which runs along all the LSTM units, and a set of gates (input, forget, and output gates) that regulate the flow of information inside the LSTM unit. These gates determine what information to keep or discard during the sequence, enabling LSTMs to capture long-term dependencies[32].

The cell state acts as a ”conveyor belt” carrying information with minor linear interactions, making it easier for information to flow across the sequence unaltered. The input gate determines what new information will be stored in the cell state, the forget gate decides what to forget from the current cell state, and the output gate determines what the next hidden state should be [32].

The diagram Figure 2.3, showing the structure of a Long Short-Term Memory (LSTM) network, includes the following components:

- Input Layer (Blue): This is where the LSTM network receives input.
- LSTM Layer (Green): This is the core part of the LSTM network. It includes the Forget Gate (which decides what information to discard from the cell state), the Input Gate (which updates the cell state with new information), the Cell State (which carries the network’s internal state from one step to the next), and the Output Gate (which decides what the next hidden state should be).
- Output Layer (Red): This is where the LSTM network outputs its predictions.

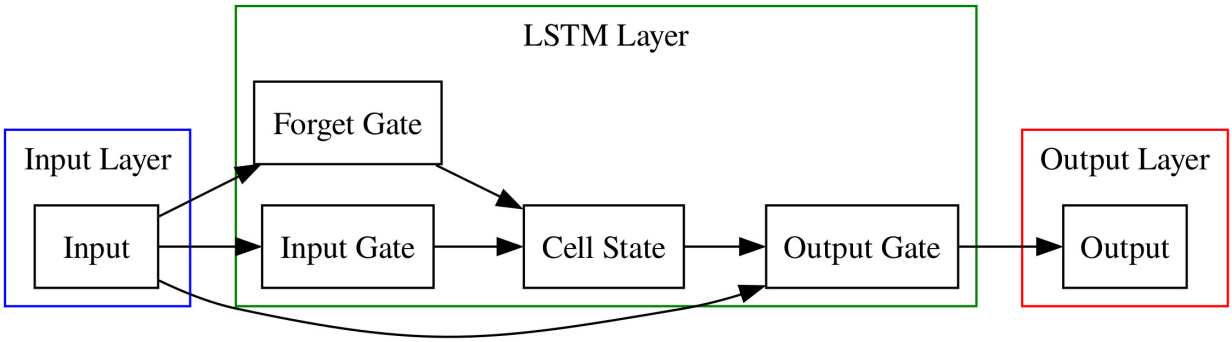

Figure 2.4: Long Short-Term Memory (LSTM) Network

LSTMs have achieved state-of-the-art performance in various sequential tasks such as language modeling, translation, and speech recognition. They can process and predict time series data, handle natural language processing tasks, and are used in many other applications where sequential data is present.

GRUs are similar to LSTM, but with a simpler architecture that combines the forget and input gates into a single ”update gate,” making them computationally more efficient, though potentially less powerful.

- **Convolutional Neural Networks (CNN):** CNNs, widely used for image analysis, handle large amounts of pixel density and greatly reduce the number of parameters, making them highly efficient. Unlike traditional ANNs, where input is represented as a feature vector, CNNs treat input as a matrix, creating a convolutional layer. A typical CNN consists of one or more convolutional layers, each connected to a respective pooling layer [32].

2.1.3.7 Unsupervised learning architectures

Unsupervised Learning Architectures in deep learning are designed to identify patterns in data without any labeled outputs to guide the learning process.

- **Autoencoders (AE):** Autoencoders are unsupervised ANNs that encode input data efficiently, a process known as latent representation or encoding [32]. They reconstruct the same data using fewer nodes than the input, making them commonly used for dimensionality reduction. Autoencoders can be used to learn a representation of input data while ignoring noise [64].
- **Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN), and Generative Adversarial Networks (GAN):** RBM, DBN, and GAN are more advanced unsupervised learning architectures. RBMs and DBNs are used for feature extraction and representation learning, while GANs consist of two neural networks contesting with each other in a game-theoretic framework [32].

2.1.3.8 Reinforcement learning architectures: DRL:

Unlike supervised and unsupervised learning, Reinforcement Learning (RL) problems are formulated as discrete-time stochastic processes. The learning process interacts with the environment via an iterative sequence of actions, state transitions, and rewards, aiming to maximize the cumulative reward [28] [64]. Deep Reinforcement Learning (DRL) combines deep learning and reinforcement learning, handling large state spaces, and function approximation [28].

2.1.3.9 Evaluating deep learning models

The evaluation of deep learning models is just as crucial as in any machine learning project. The primary purpose of model evaluation is to estimate how well a model will perform on unseen data. It also helps in comparing the performance of different models or different configurations of the same model.

The process typically involves splitting the available dataset into a training set and a test set, where the training set is used to train the model and the test set is used to evaluate the model’s predictive accuracy. Another popular method of evaluation is cross-validation, especially K-fold cross-validation, where the dataset is divided into ’K’ subsets and the model

is trained 'K' times, each time using a different subset as the test set and the remaining subsets as the training set.

Performance measures vary based on the type of problem at hand. For classification tasks, common metrics include accuracy, precision, recall, F1 score, and area under the ROC curve (AUC-ROC). For regression tasks, mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R^2 score are often used. Table 2.2 from before, summarizes the evaluation metrics mentioned above.

It is also crucial to examine the learning curves during training, which can provide insights into whether a model is underfitting, overfitting, or well-fitted to the training data. This information can guide adjustments to the model's complexity or the amount of training data needed.

2.1.4 Deep learning in stock market prediction

Deep Learning (DL) techniques have been extensively adopted in stock market predictions due to their ability to extract complex patterns and make sense of large volumes of data. However, when applying these techniques to the stock market, some unique considerations arise, ranging from model's composition, backtesting, to specific evaluation requirements and criteria. Notably, these factors extend beyond the traditional machine learning (ML) framework but are critical for financial models given the potential monetary implications [64].

2.1.4.1 Applications

DL models deployed in the stock market must consider specific factors related to financial data that surpass typical bias-variance tradeoffs observed in other applications. Unique aspects like sampling intervals, stationarity of financial data, and backtesting methods require special attention. For instance, financial time-series data are non-stationary, meaning their statistical properties (like mean, variance, covariance) change over time. Therefore, models must be capable of adapting to such data dynamics. Techniques such as differencing and fraction differencing are employed to transform the non-stationary financial data into stationary time series, enhancing the model's performance [64] [22].

2.1.4.2 Backtesting

The role of backtesting in stock market prediction models is of paramount importance. Unlike typical ML models where splitting data into training and testing sets to evaluate performance suffices, financial models necessitate the evaluation of the model's profitability and risk volatility [3]. Backtesting involves the simulation of trading strategies using historical data to evaluate the model's performance, aiding in discarding unfit models and mitigating selection bias [3].

To effectively conduct backtesting, unbiased and representative data is required, preferably spanning different periods or an adequately long timeframe. Walk-forward backtesting is a common approach, simulating trading actions using historical data in chronological time. Though this method does not assure future performance, it facilitates system evaluation based on past performance [22].

However, backtesting should be conducted with integrity, keeping in mind the risk of backtest overfitting. It is not advisable to excessively fine-tune an algorithm in response to specific events that might impact performance. Using diverse historical data for backtesting can help avoid misinterpretations and improve the model's generalizability [22].

2.1.4.3 Evaluation

A crucial aspect to discuss involves the evaluation metrics. In most machine learning applications, metrics such as accuracy and precision are used to measure the algorithm's predictive ability. However, when applied to the financial market, what is ultimately gauged is the algorithm's performance with respect to returns or volatility. It is hence crucial to demonstrate consistency across different financial evaluations of models and strategies, a principle previously emphasized in the context of avoiding overfitting during backtesting [64].

The diagram Figure 2.5, categorizing the different evaluation methods, includes the following components:

- Evaluation Metrics: This is the broadest category, encompassing all the metrics used to evaluate a model's performance.
- Financial Metrics: These are metrics specifically designed to evaluate the performance of models in the financial market. They include among others, Returns, Volatility, Sharpe Ratio, Sortino Ratio, Maximum Drawdown (MDD), Calmar Ratio, and Value at Risk (VaR).
- Traditional Machine Learning Metrics: These are metrics derived from traditional machine learning applications. They include metrics derived from a confusion matrix and metrics calculated as the difference between the predicted and observed target values.

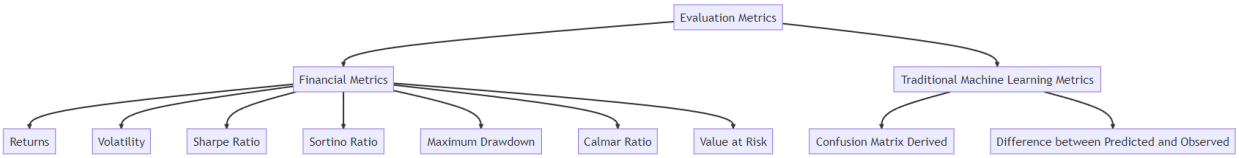

Figure 2.5: Evaluation Methods

Returns is a financial evaluation metric that measures the profitability. It is typically measured as a rate over a specific window of time, such as a day, month, or year. Returns may also be annualized over several years, a method known as the Compound Annual Growth Rate (CAGR). Higher returns over comparable time windows indicate better model performance [64].

However, focusing solely on returns could offer an incomplete picture of a model's performance. Therefore, it's important to consider volatility as well. Volatility measures the degree of variation in the price of an asset over a specific timeframe. Like returns, volatility is often reported on daily, monthly, or yearly bases. Contrary to returns, lower volatility is indicative of a better model performance [64].

Moreover, to gain insights from both returns and volatility, the Sharpe ratio is often employed. The Sharpe ratio allows investors to identify low-risk investments by comparing investment returns with risk-free assets such as treasury bonds. It measures the average returns, accounting for risk-free assets per unit of volatility. A higher Sharpe ratio signifies better model performance. However, it's worth noting that the Sharpe ratio assumes a normal distribution of the data, which might not be the case for all financial data. To account for this, the Sortino ratio is sometimes used, which only considers the standard deviation of the downward price movement [64].

Other commonly used financial metrics include the Maximum Drawdown (MDD) and the Calmar ratio, both instrumental in assessing the risk involved in an investment strategy. MDD describes the largest loss from a peak to a trough in an investment, indicating past investment losses. Lower MDD signifies a better strategy, with a zero value indicating no loss in investment capital. The Calmar ratio, on the other hand, measures the MDD-adjusted returns on capital to evaluate an investment strategy's performance. A higher Calmar ratio indicates a superior strategy [64].

Further, the Value at Risk (VaR) metric is considered essential in the reviewed works. VaR measures risk exposure by estimating the maximum loss of an investment over time, using historical performance [64].

Despite the focus on financial metrics, traditional machine learning metrics based on model prediction accuracy are still relevant and commonly used. These metrics are either derived from a confusion matrix or calculated as the difference between the predicted and observed target values. Hence, while the choice of metrics can vary, it's crucial to adopt metrics that effectively capture the nature of financial markets and the specific requirements of a model's performance [64].

2.1.4.4 DL models for financial time series forecasting

Statistics [80], find that the Long Short-Term Memory (LSTM) model was the most favored by researchers for financial time series forecasting. LSTM, along with its variations, employs time-varying data with embedded feedback representations, which has shown to enhance performance in time series prediction tasks. This is especially beneficial since most financial data encompass time-dependent components, making LSTM an ideal choice for financial time series forecasting [80].

LSTM is derived from a more general family of models, the Recurrent Neural Networks (RNNs). A considerable portion of published papers for time series forecasting falls into the RNN model category, with LSTM being the predominant choice. Given the ordinal nature of data representation, researchers often consider RNN, Gated Recurrent Unit (GRU), and LSTM as viable options for their model choices [80].

The Deep Multilayer Perceptron (DMLP) has also seen application in time series forecasting due to its market dominance and acceptance in the Machine Learning community. However, there is a fundamental distinction between the application of DMLP and RNN-based models for financial time series prediction. DMLP is suitable for both regression and classification problems, but the data's order independence must generally be preserved to better utilize the internal dynamics of such networks [80].

Recently, Convolutional Neural Networks (CNNs) have gained popularity in this field. CNNs work well for classification problems and static data representations. The use of CNNs has seen innovative transformations of 1-D time-varying financial data into 2-D

image-like data, thus leveraging the power of CNNs [80].

Other models including Deep Belief Networks (DBN), Autoencoders (AE) and Restricted Boltzmann Machines (RBM) have also been used, with superior performances reported in some cases [80].

2.2 Understanding financial market and predictions

The world of financial markets is vast and complex, with numerous elements and intricate mechanisms at play. At its core, a financial market is a marketplace where buyers and sellers trade assets such as equities, bonds, currencies, derivatives, and commodities. It is a platform that facilitates the buying and selling of these assets, contributing to the smooth operation of the economy [42].

The prediction of financial markets is an area of considerable interest, given the enormous economic implications. Predicting the trends of the market, whether it's a rise or fall, allows for strategic planning and investment. However, due to the many variables involved, predicting these trends is anything but straightforward [91][16]. The process involves statistical modeling, machine learning, and deep learning techniques to analyze historical data and forecast future market trends [81].

2.2.1 Understanding the stock market and significance

Transitioning into a closer examination, the focus is turned towards the stock market, a paramount element of the financial markets. Rooted deeply within modern economies, the stock market stands as a critical measure of a country's economic vitality. It represents a crucial part of a country's economic health and is a significant indicator of the existing business conditions and future economic prospects [40]. It provides a platform where publicly held companies can raise capital from investors who, in return, receive a share of the company's ownership [90].

The stock market's significance is twofold. For companies, it offers an opportunity to raise necessary capital for expansion or debt repayment. For investors, it provides a chance to share in the profits of the companies by becoming partial owners [90]. The dynamism and continuous evolution of the stock market make it a vibrant and a challenging area for research and prediction.

2.2.2 What is a stock?

A stock represents a fractional ownership in a company and establishes a claim on a proportion of the company's assets and earnings. There are two main types of stock: common and preferred. Both forms serve as a source of equity capital for a firm, yet preferred stock, bearing similarities to debt, is significantly different from common stock [30].

Common stock is the most commonly issued type of stock by companies. The true owners of a corporate business are the common stockholders. Holders of common stock exercise control by electing a board of directors and voting on corporate policy. However, common shareholders are on the bottom of the priority ladder for ownership structure [30].In the case of liquidation, common shareholders have rights to a company's assets only after bondholders, preferred shareholders, and other debt holders have been paid in full [73].

On the other hand, preferred stockholders receive dividends before common shareholders and have superior claims on company assets. Dividends, portion of a company's profits that are distributed to its shareholders, are usually guaranteed for preferred shares, with these shareholders typically devoid of voting rights. [30]. Preferred stock pays a cash

dividend expressed in terms of dollars per share and takes precedence over common stock in terms of dividend payments and asset distribution during liquidation [73].

Stocks principally serve to allow investors a way to participate in a company's financial performance. As the company's value grows, so does the value of the stocks held by investors. Stocks are bought and sold predominantly on stock exchanges, and they are a key ingredient for individual retirement accounts, mutual funds, and other types of investment accounts [30].

The inherent volatility in stock investments can be significant and presents risks, but the potential for high returns over the long term also exists. Therefore, understanding the nature and function of stocks is crucial for making informed investment decisions and creating a balanced, diversified investment portfolio [11].

The price of a stock is a manifestation of the market's appraisal of the company's present value and forecasted growth. This price is in constant motion during trading hours, moving in response to supply and demand dynamics, and influenced by multiple factors such as company news, earnings reports, geopolitical events, and shifts in market sentiment [70]. Market sentiment is driven by various factors such as investor psychology, macroeconomic indicators, and significant events affecting the global economy [8]. These unpredictable elements gives rise to the frequent fluctuation observed in stock prices.

Stock market trading encompasses three major stages: stock selection, purchase of a quantity of shares, and eventual selling for profit. The timing to buy and sell is extremely crucial. A selling rule can be specified by two preselected levels: a target price and a stop-loss limit [103].The bid and ask prices, representing what a dealer is willing to pay for a security and what they wish to receive for it respectively, determine the bid-ask spread, the dealer's profit [73]. Traders and investors use various strategies and techniques to decide when to buy and sell stocks. These strategies could be based on a fundamental analysis of the company, or a technical analysis of the stock's price movements and trends, or a combination of both [63]. The objective of trading is to yield profit from the price movements in stocks.

Stock exchanges play a pivotal role as the principal venues for the buying and selling of stocks. Prominent exchanges like the New York Stock Exchange and the London Stock Exchange are cornerstones of global finance, facilitating regulated and transparent trading activities.

This is a simplified overview of what a stock is. The world of stocks is complex and multifaceted, encompassing many different types of securities, investment strategies, and market dynamics. Throughout this thesis, the focus will remain on common stocks, as these are the most relevant for the subsequent analysis and discussion.

2.2.3 Stock market data

Stock market data provides a snapshot of market activity and is an essential resource for traders and investors. It gives valuable insights into market trends and assists in forming predictions about future market movements. Traditional stock market data includes trading activity such as opening, closing, high, and low prices. However, with advancements in technology, the scope of data has expanded to include a variety of other data types. This broad spectrum of data types, each with unique attributes, enhances the accuracy and comprehensiveness of market predictions.

The process of market prediction involves collecting relevant data as the foundation. This can either be based on historical prices or external data sources that have an impact on the stock market. In the efficient-market hypothesis, it is believed that asset prices already encompass all available information. However, this conclusion is often contested in practical scenarios. Hence, a mix of different data sources is commonly employed in stock market predictions [41]. For example, some studies have combined market data with other forms of data, technical indicators, Wikipedia traffic, Google news counts, and generated features for their predictions [96], and others incorporated fundamental data, knowledge graph data, and news into their prediction models [13] [41].

Stock market prediction involves more than just the collection of raw data. Based on various combinations of data sources, prior studies have examined the use of deep learning models for predicting stock market prices and movements. In general, a typical workflow encompassing four steps is adopted by most studies. This includes Raw Data, Data Processing, Prediction Model, and Model Evaluation. Each of these steps plays a crucial role in establishing a reliable and reproducible methodology for future research in stock market prediction [41].

2.2.3.1 Raw data

The initial step in predicting stock market movements is the collection of raw data. This data forms the basis of any prediction model and can be either intrinsic, like historical prices, or extrinsic, like various data sources affecting the stock market. Intrinsic data is based on the assumption that history repeats itself. In contrast, extrinsic data comprises various sources, each contributing differently to the market prediction. The array of data sources used is a testament to the complexity of stock market predictions, indicating that relying solely on intrinsic data may not suffice. Thus, incorporating a variety of extrinsic data sources ensures the development of more robust and accurate prediction models [41].

Types: Various forms of raw data are used for stock market prediction, and these can be broadly classified into seven categories [41]:

- **Market data**

  Market data consists of all the recorded trading activities in a stock market, including various metrics like opening, closing, highest, and lowest prices, trading volumes, etc. This data is used as both the input and the predictive target in many models.

- **Text data**

  This data source comprises text inputs from individuals like social media posts, news articles, and web search results. Despite the complexity in collection and processing, this data can yield valuable insights that aren't directly captured by market data. Sentiment analysis is a common technique applied to text data to create a sentiment factor for predictions.

- **Macroeconomics data**

  This type of data gives a snapshot of the economic conditions of a specific region, country, or sector, offering key indicators such as the Consumer Price Index (CPI)

and Gross Domestic Product (GDP). These indicators provide insight into the overall health of the stock market and offer context for market trends.

- **Knowledge graph data**

  This data shows the connections between different companies and markets. For example, stocks from the same sector may be influenced by the same news events. The utilization of open-source knowledge graph data can significantly improve prediction outcomes.

- **Image data**

  Following the advancements of convolutional neural networks in 2D image processing, candlestick charts are now being used as input images for stock predictions. Still, the use of images or videos to keep track of companies' activities is less common due to cost and privacy concerns.

- **Fundamental data**

  This refers to accounting data that is typically reported on a quarterly basis, such as assets, liabilities, etc. Despite its infrequent reporting and potential inaccuracies, this type of data can offer valuable financial insights.

- **Analytics data** Analytics data is derived from detailed reports generated by investment banks and research firms. These reports, which offer in-depth analysis of companies' activities, competitions, and business models, can be useful for making predictions, though they can be expensive and often shared among multiple users [41].

Length / period: The choice of data length is a crucial aspect when assessing the performance of various prediction models. Using data from a short time span can lead to overfitting, while using data from a longer time span may cover diverse market conditions and risk presenting obsolete results. Data availability and cost also come into play when choosing the data length. Intra-day data of high quality tends to be more expensive, and most studies using such data typically consider a time period of less than a year. The term 'lag' refers to the period of input data used by a model. For instance, a 30-day lag in daily prediction means the model uses data from the past 30 days to construct the input features. 'Horizon' refers to the future period that the model aims to predict. The majority of studies focus on short-term prediction horizons, with only a few exceptions targeting longer horizons [41].

2.2.3.2 Data processing

Data processing forms an integral part of the workflow for stock market prediction. During this phase, the raw data collected undergoes a series of operations to prepare it for analysis and model building. The operations typically involve dealing with missing or corrupted data and reducing noise in the data to enhance its interpretability and reliability. These steps are crucial to improving the accuracy of the predictions made by the models [41].

Missing and corrupted data: Despite the general reliability of market data, missing data is an inevitable occurrence, particularly when aligning data types with differing sampling frequencies. For instance, market data, which usually has a high sampling frequency, might need to be combined with fundamental data that has a lower frequency.

To resolve this, a technique called forward imputation is employed. It involves propagating the last valid observation forward to the next valid point. This approach ensures that data integrity is maintained, while also preventing the leakage of future information, that could bias the prediction model [41].

Denoising: In an era of instant information and ubiquitous internet access, the stock market is increasingly influenced by an endless stream of news updates and analysis. With countless devices, news outlets, and social media platforms providing a second-by-second account of various developments, the prevalence of ”noise” has become a notable issue in stock market data. This noise is a fundamental component of these constant updates, particularly dominating over short periods of time. Many ”breaking news” stories may seem more exciting and crucial than they truly are due to this noise, which can lead to market participants overreacting to real-time news [92].

The process of stock trading can often be clouded with irrational behaviors, introducing noise into the market data. This noise can distort the true trend of price changes and potentially mislead the prediction process [41]. However, it is essential to note that the longer the horizon period, the more information we have, and the higher our prediction’s resistance to noise [83]. Therefore, it’s important to implement de-noising techniques to filter out these discrepancies and ensure the data accurately represents the market dynamics.

Several techniques have been utilized in previous studies for this purpose. The wavelet transform, a signal processing method, has been employed to eliminate noise in stock price time series [10]. Additionally, another study utilized kNN-classifiers based on two different training sets to discard noisy data in the data preparation layer [88].

Feature correlation and extraction: Feature correlation and extraction are two critical aspects of pre-processing data for machine learning and deep learning applications. Feature correlation is the statistical relationship between two variables or features in a dataset. In contrast, feature extraction involves reducing the dimensionality of the dataset by identifying the most relevant features or creating new features from the existing ones.

- **Feature Correlation**

  Feature correlation, according to Mark A. Hall’s 1999 study ”Correlation-based Feature Selection for Machine Learning,” is a critical aspect of supervised machine learning. His research emphasizes the importance of using feature correlation for selection, arguing that this can help remove irrelevant, redundant, and noisy features. The correlation-based feature selection (CFS) algorithm evaluates the merit of feature subsets based on their predictability and redundancy [35].

  The correlation measure in the CFS algorithm, such as symmetrical uncertainty or minimum description length (MDL), should prefer predictive features and penalize redundancy. This correlation-based feature selection can enhance the performance of machine learning algorithms and reduce the number of features used in learn-

ing. However, it might struggle in datasets containing strong feature interactions or features predictive in small areas of the instance space [35].

Moreover, there are other correlation techniques, such as cross-correlation and autocorrelation [83]. Cross-correlation is a measure of similarity between two signals, often used to find features in an unknown signal by comparing it to a known one. On the other hand, autocorrelation measures the correlation of a signal with a delayed copy of itself. Both techniques are commonly used for signal processing to detect patterns or periodic signals in the data.

- **Feature Extraction**

  Feature extraction, as defined by Jiang in ”Applications of Deep Learning in Stock Market Prediction: Recent Progress,” is the process of deriving relevant input features from raw data based on domain knowledge [41]. Technical analysis is an example of feature extraction applied to market data, where various indicators forecast price directions based on historical prices and volumes.

  As per Jiang, tools for feature extraction from text data have seen significant progress over the past few years, owing to various deep learning models developed for natural language processing. Techniques like bag-of-words (BoW), word2vec, and Global Vectors for Word Representation (GloVe) are used for text data. These models convert each word in a document into a vector of real numbers, capturing the semantic meaning of words in a high-dimensional space [41].

  Apart from text, knowledge graph data, representing data with graph relational models, has been used for feature extraction. The TransE model is one example, which represents one-to-one relationships in a computationally efficient way. All these feature extraction techniques play a pivotal role in enhancing the performance of machine learning models [41].

The successful application of machine learning and deep learning in stock market prediction requires a careful approach to feature correlation and extraction. By correctly correlating features and extracting the most meaningful ones, we can build models that more accurately and efficiently predict market dynamics.

Dimensionality reduction: In domains with high-dimensional data, such as the financial market, dimensionality reduction is vital, especially when many features might be highly correlated. Such an example is the technical indicators derived from historical open, high, low, close prices, and volume. To counteract the overfitting problem in deep learning models, it’s necessary to apply dimensionality reduction techniques that can transform this high-dimensional input data into a lower-dimensional space, enhancing computational efficiency and model generalization [41].

One widely used dimensionality reduction technique is Principal Component Analysis (PCA). PCA uses Singular Value Decomposition to project the input data onto a lower-dimensional space. This transformation can help mitigate the impact of noise in the data. For instance, Zhong & Enke (2017) applied PCA to improve the prediction accuracy of their model for predicting the daily direction of SPY for the next day [41] [104].

Apart from PCA, there exist several other techniques for dimensionality reduction, such as Independent Components Analysis (ICA), autoencoders, and Empirical Mode Decomposition (EMD). Huang et al. (2018) utilized an approach named Sub-mode Coordinate

Algorithm (SMC) to tackle this challenge [39]. They first integrated multi-sourced data using a tensor and then proposed an improved SMC model to reduce the variance of their subspace in each dimension produced by tensor decomposition [41].

Feature selection is another strategy for dimensionality reduction, aiming to identify a subset of input features that are most relevant for the prediction task. Techniques like the Chi-square method and Maximum Relevance and Minimum Redundancy (MRMR) feature selection are commonly employed. The Chi-square method is used to establish the dependence of the target variable on the predictor variable, while MRMR aims to select features that are most relevant to the target variable while being least correlated with each other [41].

Overall, dimensionality reduction, whether achieved through data transformation techniques such as PCA and ICA, or feature selection methods like the Chi-square and MRMR, is crucial in dealing with high-dimensional data, improving model performance, and enhancing computational efficiency in machine learning tasks.

Feature normalize and standardize: With varied scales of input features in datasets, the processes of feature normalization and standardization are crucial for improving machine learning model performance and training speed. These preprocessing techniques help ensure that machine learning models can effectively learn from the data without being influenced by the scales of the different features [41].

Feature normalization is the rescaling of input features to a specific range. In most cases, the data is rescaled to fall between 0 and 1, ensuring that each feature contributes evenly to the model's performance. This process is often achieved by subtracting the minimum value of the feature and then dividing by the range of that feature. In some contexts, normalization is performed to scale the data between -1 and 1, particularly when the data distribution is not uniform or contains outliers [41].

Feature standardization, on the other hand, involves rescaling the distribution of features so that they have a mean of 0 and a standard deviation of 1. A common method used for standardization is the z-score method, which subtracts the mean of the feature and divides by its standard deviation. This process results in a standardized distribution of features where the majority of the values lie within a small range around 0. Standardization is especially beneficial when the algorithm is sensitive to the magnitude of the features, such as in the case of distance-based models [41].

Normalization and standardization are techniques applied based on the distribution and specific needs of the data. Normalization, which rescales values to a range of 0 to 1, is ideal for data that doesn't follow a Gaussian distribution or when feature boundaries are known. Conversely, standardization, which transforms data to a mean of 0 and standard deviation of 1, is preferred when the data follows a Gaussian distribution, and is particularly useful in machine learning algorithms like support vector machines and linear discriminant analysis that assume normally distributed input data [60]. By scaling the features to a comparable range, these techniques help improve the speed and stability of the learning process, thereby enhancing model performance.

Data split: In machine learning and deep learning fields, the evaluation of prediction models typically involves splitting the data into different sets, namely, in-sample/out-of-sample or train/validation/test sets. The model is trained on the training set or in-sample

data, while the hyper-parameters are optionally fine-tuned on the validation set. The final performance of the model is then evaluated on the test set or out-of-sample data [41].

For further refinement, k-fold cross validation is often applied. This technique involves splitting the dataset into k consecutive folds, with the model trained on k-1 folds and tested on the remaining fold. When it comes to time series tasks such as stock prediction, a rolling (or sliding, moving, walk-forward) window for the train-validation-test split is often used [10] [17]. This method involves using only the most recent data samples for each new training round of the prediction models. Alternatively, successive training sets can be used, which are the union sets of the rolling training set that came before them [41].

2.2.4 Stock market prediction

Stock market prediction primarily focuses on determining the future direction of a financial exchange's stock value. Accurate prediction of these movements can lead to significant profits, making this a highly attractive field of study. Yet, it is equally acknowledged as a complex task, owing to the multiple factors contributing to the volatile nature of stock markets, including interest rates, politics, and economic growth [1].

Two primary approaches dominate the field: technical analysis and fundamental analysis. Technical analysis utilizes historical stock price data to forecast future values, while fundamental analysis leans heavily on unstructured textual information like financial news and earnings reports. An emerging emphasis on the latter stems from the increasing volume of valuable market information available publicly online. The application of text mining strategies to extract crucial information from such data can enrich market behavior analysis, enhancing prediction capabilities [1].

2.2.4.1 Unpredictability of the stock market

The complexity and volatility of the stock market have led some to consider its movements as fundamentally unpredictable. Two leading theories that uphold this perspective are the Random Walk Hypothesis (RWH) and the Efficient Market Hypothesis (EMH) [63].

- **The random–walk hypothesis (RWH)** The RWH proposes an unflattering view of stock market predictability. It posits that stock prices are fundamentally stochastic, which renders any attempt to predict their future movements destined to fail. If the market is truly stochastic, then the possibility of predicting it reliably may be negligible [63].
- **The efficient market hypothesis (EMH)** The EMH, put forth by Fama in 1965, suggests that the stock market is "informationally efficient". It argues that the market excels at determining the correct price for stocks [25]. However, Fama later revised the EMH, dividing it into three levels of efficiency: weak-form, semi-strong, and strong [24]. This fragmentation of the hypothesis has led to debates over its validity and the degree of market efficiency, if any [63].

Despite the inherent unpredictability implied by these hypotheses, an understanding of historical stock data and fundamental or financial data about a company may still allow for successful prediction of its future stock prices. Thus, while the stock market's movements may appear random, there is an element of predictability accessible to those with both fundamental and technical knowledge of the market.

2.2.4.2 Types of stock forecasting

Stock market prediction remains a topic of widespread interest due to its complex and potentially profitable nature. An array of prediction types exist within the domain, each possessing their own distinct characteristics and methodologies, which are employed in attempts to analyze and predict various aspects of the financial markets. Each type of forecasting serves a unique purpose, with different methods and models used depending on the specific aspect of the market being forecasted [80].

- **Equity stock price forecasting** Equity stock price forecasting, which is the prediction of the price of a given stock, is one of the most frequently researched and implemented areas within financial forecasting. As the most commonly studied application of financial time series forecasting, equity stock price forecasting is pivotal in investment decisions and strategic planning. Multiple variables and parameters are considered in these predictions, from high-frequency trading and intra-day price movements to longer-term inputs such as daily, weekly, or monthly stock closing prices. Moreover, a variety of data sources are harnessed in the creation of predictive models, including raw price data, technical and fundamental analyses, macroeconomic data, social media feeds, investor sentiment, and more [80].

  Despite the inherent unpredictability of stock prices, as suggested by the Random Walk Hypothesis, efforts have been made to glean insights into price movements by incorporating various influencing factors alongside historical prices. Given the temporal nature of price changes, recurrent neural networks (RNN) and long short-term memory networks (LSTM) are often used in these predictive models. Recent innovations in this field have seen the introduction of hybrid models, which combine different types of neural networks to harness their respective strengths [64].

- **Index forecasting** Index forecasting has garnered considerable attention in the realm of finance and economic research, eschewing the focus from individual stock prediction to the broader prediction of stock market indices. Owing to their composition from diverse sectors, these indices exhibit less volatility compared to individual stocks and are representative of the overall market trends and economic state. Researchers have utilized various global indices for these experiments such as the S&P500, NIKKEI225, DJIA, HSI, SZSE, and more [80].

  Diverse methodologies have been employed for index prediction. Some studies utilize raw time series data alone, while others incorporate additional factors like technical indicators, index data, social media feeds, news, and various statistical features of data. For instance, deep learning models like MLP, RNN, LSTM, and DNN have been prevalent in index forecasting [80].

- **Commodity price forecasting** Commodity Price Forecasting refers to the practice of predicting the price of commodities such as gold, silver, oil, and copper. Given the increased availability of commodities for public trading through online exchanges, interest in this area is expected to grow in the years ahead. Different methods and models, including DNN, RNN, FDDR, and CNN, have been utilized for forecasting commodity prices [80].

- **Volatility Forecasting** Volatility forecasting deals with predicting fluctuations in the price of an asset over a specified time period, which is crucial for risk assessment and asset pricing. Various models and methods have been implemented by researchers

for accurate volatility forecasting, including LSTM, RNN, CNN, MM, and Generalised Auto-Regressive Conditional Heteroscedasticity (GARCH) models [80].

- **Currency Forecasting (FOREX)** Currency Forecasting, also known as Forex forecasting, is a critical area of finance due to the immense volume of transactions it represents. Operating 24/7, the foreign exchange market witnesses trillions of dollars worth of transactions daily, making it the largest financial market in the world. Numerous online platforms facilitate leveraged trading in this market, contributing to an intense interest in profitable trading strategies. This focus has led to a proliferation of research on forecasting forex using deep learning (DL) models [80].

  The dominant currency in most financial transactions is the US Dollar, thus a majority of forex prediction research includes USD. Other currencies and models are used depending on regional differences and the research focus. Techniques used for forex price forecasting include RNN, LSTM, CNN, DBN, DNN, AE, and MLP [80].

- **Bonds, ETFs, Options, derivatives price forecasting** Bonds, ETFs (Exchange-Traded Funds), options, and derivatives are crucial financial instruments. Bonds are loans investors give to entities like governments or corporations in exchange for periodic interest payments and the return of the bond's face value when it matures. ETFs are investment funds traded on stock exchanges, mimicking the performance of a specific index, sector, commodity, or asset. Options grant the buyer the right to buy or sell an underlying asset at a specific price before a certain date, without obligation. Derivatives are contracts whose value stems from one or more underlying assets, which can include stocks, bonds, commodities, currencies, interest rates, and market indexes.

  Forecasting the prices of these instruments is complex, particularly due to their intricate nature and reliance on various factors like market volatility, interest rates, and the underlying asset's price. For bond price prediction, which is often seen as an indicator of economic health, deep learning approaches have been explored, but not extensively [80]. Deep learning methods for forecasting prices of ETFs, options, and derivatives are also being studied, although these implementations are less common.

- **Cryptocurrency forecasting** Cryptocurrency Price Forecasting has emerged as a popular area of study in recent years due to the meteoric rise and volatile nature of cryptocurrencies. The most notable of these, Bitcoin, saw its value skyrocket from $1000 USD in January 2017 to $20,000 USD in January 2018, capturing not just the attention of the financial world but also the general public. These fluctuations have led to a surge in research focusing on price prediction and trading strategy development for Bitcoin and other cryptocurrencies [80].

  Various methods, such as DNN, LSTM, GRU, RNN, and traditional methods like ARMA, ARIMA, ARCH, and GARCH, have been used in cryptocurrency price forecasting. Given the intrigue around cryptocurrencies and the technology that underpins them, it's likely that many studies will continue to emerge in this field [80].

- **Trend forecasting** Trend forecasting differentiates from price forecasting by predicting the direction of asset price movements instead of the exact prices. This shifts the problem from regression to classification, altering the corresponding performance metrics. Various methodologies have been used, categorized broadly into those using only raw time series data, those incorporating technical indicators, price data, and fundamental data concurrently, and those employing text mining techniques or

other diverse datasets. Several models such as ANN, DNN, FFNN, LSTM, RNN, and Probabilistic NN are commonly used [80].Despite the broad range of methodologies, all approaches in trend forecasting ultimately aim at understanding the direction of asset price movements, offering a rich area for future research and development.

- **Risk management** The field of risk management, integral to maximizing financial returns by minimizing potential risks, has not been the central focus of many studies in comparison to other financial disciplines. This trend may shift due to recent global events such as the 2020 market crash prompted by the COVID-19 pandemic. These events underscore the critical need for robust risk management strategies, thereby likely spurring renewed interest and research in this area [64].

  In the context of machine learning, various methods such as feedforward neural networks (FFNN), temporal convolutional neural networks (CNN), and long short-term memory (LSTM) models can be employed for risk management. They can be utilized to estimate critical financial thresholds such as the Value at Risk (VaR), which represents the level of financial risk within a firm or investment portfolio over a specific time frame. Predicting breaches of this threshold can help in the early detection of potential shifts in market trends, from bull (rising) to bear (falling) markets, thus allowing for timely risk mitigation strategies [64].

- **Portfolio management** Portfolio management, a crucial field in finance, has seen the increasing application of deep reinforcement learning (DRL) techniques. It involves determining the optimal mix and proportions of various assets in a portfolio to maximize returns and minimize risks. Researchers have experimented with various DRL algorithms for portfolio management, including Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), and Policy Gradient (PG). These methods, although promising, often face challenges such as the need for extensive data, especially during a bull market, and the complexity of mitigating risks. Other models, such as Q-Learning, have been used for optimal portfolio management across multiple assets, using a Markov Decision Process (MDP) to simulate trading actions and derive practical strategies [64].

2.2.4.3 Prediction methods

Predictive methods for stock market behavior generally fall into three broad categories: fundamental analysis, technical analysis, and a combined approach using both qualitative and quantitative data. The choice of prediction method often depends on the type of data that a financial analyst or a machine learning model is equipped to handle, as well as the specific goals of the forecasting process. It's essential to recognize that these methods are not mutually exclusive, and in many scenarios, they are used in conjunction with each other to generate more accurate and robust predictions [63].

- **Fundamental analysis (qualitative data)** Fundamental analysis, primarily based on qualitative data, is a method that involves examining financial reports, investor sentiments, and macroeconomic variables that may influence a company's stock price. This type of analysis looks beyond the numerical metrics of a company and considers broader elements such as the overall economic environment, industry trends, and company management. Although fundamental analysis has been used less frequently than technical analysis, it has shown promising results, especially when sentiment analysis of social network sites is used as a predictor of market

movement. However, despite the documented positive correlation between macroeconomic variables and stock market returns, this aspect of fundamental analysis is less explored, leaving room for further research [63].

- **Technical analysis (quantitative data)** Technical analysis, on the other hand, is based on quantitative or structured data, particularly historical stock prices. This type of analysis assumes that all pertinent information is already reflected in the stock's price. The idea is that past trends and patterns in a company's stock price can help forecast future performance. Various statistical and computational tools, such as moving averages and other technical indicators, are frequently used to analyze and interpret this data, providing insights into potential trends and price movements. [63].
- **Combined (qualitative and quantitative data) analysis** A combined approach integrates both fundamental and technical analyses, leveraging both qualitative and quantitative data to improve the accuracy of stock price prediction models. A smaller subset of studies explored the combined approach, utilizing multiple data sources to encapsulate a more complete view of market dynamics. Each method, with its own unique strengths, can contribute to a more comprehensive understanding of stock market trends, and combined, there is a promising avenue for future research. [63].

2.2.4.4 Approaches

In the context of stock market analysis, the term 'approaches' refers to the different methods or techniques used to predict future stock prices. These methods are typically based on mathematical or computational models that use historical and/or real-time data to make these predictions. The choice of approach depends on various factors such as the type of data available, the investment strategy, and the prediction horizon. Different approaches have unique strengths and weaknesses and may be more or less effective depending on the specific market conditions and goals of the prediction [81].

- **Statistical Approach** The statistical approach to stock market prediction utilizes quantitative data analysis methods to identify trends or patterns in historical data, with the aim of forecasting future price movements. Traditional statistical methods include moving averages, linear regression, time series analysis, and autoregressive integrated moving average (ARIMA) models, among others. These models take into account the stochastic nature of financial time series and provide statistically sound predictions. However, the complexity of financial markets often exceeds the simplifying assumptions of many statistical models, leading to inaccuracies [81].
- **Pattern Recognition** Pattern recognition in stock market analysis is a technique that identifies recurring patterns in the historical stock market data. These patterns may indicate specific market behaviors that can be utilized to predict future price movements. Methods such as chart patterns, trend lines, support and resistance levels, and various other technical analysis indicators fall under this approach. The effectiveness of pattern recognition often depends on the length and diversity of the data set, as well as the stability of the patterns over time [81].
- **Machine Learning** Machine Learning (ML) techniques have gained significant traction in stock market prediction due to their ability to learn complex patterns from

data. ML can be further divided into supervised learning, unsupervised learning, and reinforcement learning.

  - **Supervised Learning** In this approach, a model is trained on labeled data (input-output pairs) and then used to predict outcomes on unseen data. Common supervised learning algorithms used in stock prediction include Support Vector Machines (SVM), Artificial Neural Networks (ANN), and Decision Trees (DT) among others [81].
  - **Unsupervised Learning** This involves training a model on data without predefined labels, with the aim of discovering hidden structures in the data. Techniques like clustering and dimensionality reduction fall under this category. While unsupervised learning methods are less commonly used for direct stock price prediction, they can provide valuable insights into underlying market structures [81].
  - **Reinforcement learning** Reinforcement Learning (RL) is an area of machine learning where an agent learns to make decisions by taking actions in an environment so as to maximize some notion of cumulative reward. RL has been applied in portfolio management, where the agent's goal is to maximize the total return over a certain period [81].

- **Sentimental Analysis** This approach is based on the idea that market sentiment, as reflected in news articles, social media posts, and other forms of public discourse, can have a significant impact on stock price movements. Techniques used in sentiment analysis involve Natural Language Processing (NLP), text mining, and semantic analysis. The identified sentiments can then be used as inputs to a predictive model or to guide trading strategies [81].
- **Hybrid Approach** A hybrid approach seeks to combine multiple techniques to improve the accuracy of stock market prediction. This might involve integrating statistical methods with machine learning algorithms, or combining different types of data (e.g., technical indicators, sentiment data) for input into a single predictive model. The goal of a hybrid approach is to exploit the strengths of each method while minimizing their individual weaknesses, thus enhancing the overall predictive performance [81].

2.2.4.5 Forecasting horizon

The forecasting horizon refers to the length of time into the future a prediction is made. Depending on the prediction model and its application, the forecasting horizon can vary from intraday (within the trading day), to short-term (a few days to weeks), mid-term (a few months), and long-term (more than a year). The choice of forecasting horizon depends on the investment strategy, with day traders focusing on intraday and short-term forecasts, while long-term investors focus on mid-term and long-term predictions [81].

2.2.5 Prediction evaluation

Evaluating the efficacy of prediction models in stock market analysis is crucial for determining their reliability and effectiveness. This assessment, often referred to as prediction evaluation, uses various metrics to measure the performance of models, helping users to

identify the most suitable ones for their needs. These metrics fall into four main categories: classification metrics, regression metrics, profit analysis, and significance analysis, each with its unique considerations and applications [41].

2.2.5.1 Classification metrics

Classification metrics are pivotal when predicting market movement, and is often modeled as a classification problem. The performance of these models is assessed using numerous metrics including accuracy, precision, recall, F1 score, macro-average F-score, Matthews correlation coefficient, and more. Tools such as confusion matrices and box-plots are commonly used for in-depth analysis of classification performance [41].

2.2.5.2 Regression metrics

On the other hand, regression metrics are employed when the prediction model is seen as a regression problem, such as predicting the specific stock or index price. Here, typical metrics encompass mean absolute error (MAE), root mean absolute error (RMAE), mean squared error (MSE), root mean squared error (RMSE), mean absolute percentage error (MAPE), among others. These metrics provide insights into the magnitude of the prediction errors, aiding in model refinement [41].

2.2.5.3 Profit analysis

Profit analysis is a crucial aspect of evaluation that quantifies whether the predicted-based trading strategy results in a profit. Returns, which represent changes in stock portfolio value, and risk, which can be assessed using maximum drawdown or annualized volatility, are key considerations. The Sharpe Ratio, a metric that incorporates both return and risk, is commonly used to evaluate the trade-off between them [41].

2.2.5.4 Significance analysis

Significance analysis is conducted to discover if there are statistically significant differences in predictions when comparing deep learning models to baseline models. This process, though not as commonly employed in stock prediction, is indispensable for deciding which model performs significantly better [41].

Each of these evaluation methods plays a crucial role in assessing the effectiveness and reliability of prediction models in stock market analysis. By understanding their distinct features and applications, users can select and refine models that are best suited to their predictive needs.

2.2.6 Forecasting challenges

Forecasting in the domain of stock market analysis can pose a multitude of challenges, spanning from data accessibility to the intricacies of financial models. These obstacles not only impact the accuracy of predictions, but also limit the potential advancements in research. This section describes several challenges in this field [64].

2.2.6.1 Availability of historical market data

The availability of historical market data constitutes a significant obstacle to stock market analysis. Such data is vital for building and refining prediction models, but it often comes at a high premium and is generally not readily accessible, particularly at finer levels of granularity such as intra-day and tick data. This barrier disproportionately affects research initiatives without substantial financial resources, making consistent, publicly available market data, or costly subscriptions the only viable options [64].

2.2.6.2 Access to supplementary data

Related to the prior issue is the access to supplementary data, which can greatly enhance the performance of financial models. This data includes fundamental information such as quarterly reports and alternative data such as news articles or social media discussions about the relevant company. Despite the potential for enriching predictive models, access to this data is often limited or altogether absent, representing a significant challenge for researchers [64].

2.2.6.3 Long term investment horizon

In the context of stock market predictions, the majority of studies focus on short investment horizons spanning a few days to a few months. However, a significant portion of market investments involves long-term commitments like retirement funds. Identifying young public companies with high growth potential early on can lead to sizable returns for these long-term investments. Such strategies could benefit from the use of supplementary data, underlining the need for better access to this information [64].

2.2.6.4 Effect of capital gains tax

Moreover, many studies overlook the impact of capital gains tax, especially for short-term investments where tax rates can be substantial. This omission can lead to a misrepresentation of actual returns and consequently, misleading conclusions about the effectiveness of investment strategies. Accounting for these costs is rarely done but is crucial for accurate and realistic evaluations [64].

2.2.6.5 Financial ML/DL framework

Lastly, the utilization of popular machine learning (ML) and deep learning (DL) frameworks such as scikit-learn, TensorFlow, Keras, and PyTorch is widespread in both academic and industrial research. However, there seems to be a lack of concerted effort to expand these frameworks using specialized financial works. The shortage of such frameworks designed specifically for financial ML, limits the potential for collaborative improvement and conformance to industry practices [64].

Understanding and addressing these challenges is essential for pushing the boundaries of stock market prediction research. This, in turn, will contribute to the development of more accurate and effective prediction models and strategies.

3. LITERATURE REVIEW

Extensive research is being conducted on the complex behaviors of stock markets, from traditional financial theories to the cutting-edge application of machine learning and deep learning techniques for predicting stock prices. This chapter offers an in-depth examination of the existing body of knowledge in these diverse but interconnected areas. While we have mentioned various references in the Background chapter, this Literature Review intends to delve deeper into specific techniques and their applications in stock market prediction, providing an enriched understanding of the state of the art. The following subsections will discuss Autoregressive models, Machine Learning models, Deep Learning models, Hybrid models, and Other predictive approaches. The field of stock market prediction is characterized by a vast array of research methodologies. Despite this extensive diversity, this section attempts to describe and reference several key approaches, relevant to the study.

3.1 Introduction to state of the art

Predicting the stock market has emerged as a challenging area of study. The rise of data availability and computational power has resulted in a variety of methods and models proposed and refined for stock market prediction. Ranging from econometric approaches such as Autoregressive Integrated Moving Average (ARIMA) models, to Machine Learning algorithms like Support Vector Machines (SVM) and Deep Learning techniques like Recurrent Neural Networks (RNN), various more methods have been explored. This section will provide an overview of some relevant studies in these areas.

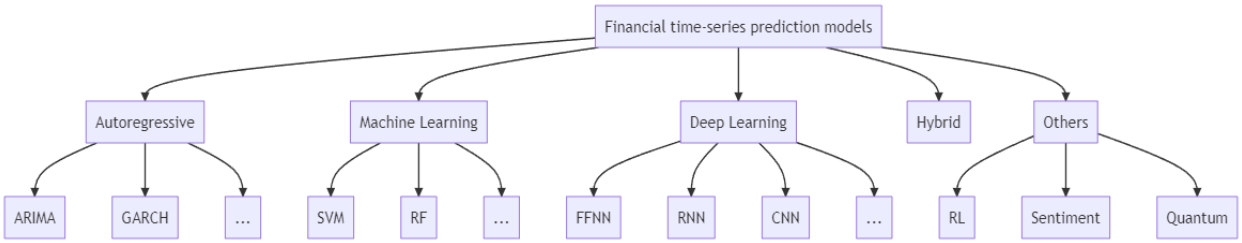

Figure 3.1: Financial Time-Series Prediction Models

Figure 3.1 presents a hierarchical representation of these various methods used in financial time-series prediction models [75]. The topmost node, ”Financial time-series prediction models”, branches into five categories: Autoregressive models, Machine Learning models, Deep Learning models, Hybrid models, and Other models. Autoregressive models subdivide into AutoRegressive Integrated Moving Average (ARIMA) , Generalized AutoRegressive Conditional Heteroskedasticity (GARCH), and others, that are statistical models that use past data to predict future data points. Machine Learning models like Support Vector Machines (SVM), Random Forests (RF), and others were applied to financial time series modeling. These models can capture complex patterns in the data and are often used for their ability to handle large datasets and high-dimensional feature spaces. Deep Learning models like Feedforward Neural Networks (FFNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and multiple other neural networks became popular for time series forecasting. These models are particularly good at capturing long-term dependencies in the data, making

them well-suited for financial time series which often have temporal dependencies. Hybrid models, combine the strengths of different types of models to improve prediction accuracy, and are also included. Lastly, the ”Others” category incorporates other novel approaches like Reinforcement Learning (RL), where a model learns to make predictions through trial and error, Sentiment analysis, which uses data from sources like news articles or social media to inform predictions, and quantum finance, which applies principles from quantum physics to finance. This broad categorization encapsulates the diversity of methods employed in financial time-series prediction.

3.2 Autoregressive models

Autoregressive models, specifically ARIMA and GARCH, have been pivotal to financial forecasting. Catering to time series data, they’ve proved to be critical in financial markets.

ARIMA, widely employed in stock market analysis, is an extension of the ARMA model, merging Auto-Regressive (AR) models and Moving Average (MA) models [53] [81]. The AR component interprets momentum and mean reversion effects observed in trading markets, while the MA component captures shock effects seen in time series [75]. Despite ARMA’s limitation of not considering volatility clustering, a common occurrence in financial time series, ARIMA mitigates this by transforming a non-stationary series into a stationary series, via finite differencing of data points, enabling more accurate forecasts [81] [93] [75].

A time series is considered stationary if its statistical properties, such as mean and variance, remain constant over time. The ARIMA model manages to separate the signal (underlying trend) from the noise in a time series, enabling forecasting based on the signal [75]. This method follows the Box-Jenkins model developed for identifying the best-fitting ARIMA model to a given time-series [93].

The GARCH model, as an extension of the ARCH model, has gained popularity in financial time series analysis due to its ability to handle volatility clustering, a common phenomenon in financial markets. It allows for varying variances over time, which can better capture the volatility dynamics in financial data [75].

Additional research demonstrates the applicability of ARIMA in different contexts. For example, Devi et al. (2013) [4] applied the ARIMA model to historical data of four Indian midcap companies, considering aspects often overlooked in the stock analysis literature such as dimensionality and the expectations of a naïve investor. They used the Akaike Information Criterion Bayesian Information Criterion (AICBIC) test to predict the model’s accuracy, finding that the Nifty Index performed best for naïve investors due to its low error and volatility [81].

Similarly, Ariyo et al. (2014) [2] illustrated the process of building ARIMA models and determining the optimal model using criteria like the standard error of regression, adjusted R-square, and Bayesian information criteria. The selected ARIMA model performed satisfactorily in predicting the stock prices of Nokia and Zenith Bank, affirming the model’s utility in stock analysis, especially for short-term predictions [81].

Literature has reported certain limitations of the ARIMA model, especially in the context of financial data that exhibit asymmetries, irregular time intervals, and high volatility. These complexities often lead to the model’s assumption of constant variance being violated. To address these issues, hybrid models, such as the integration of ARIMA and GARCH [75].

3.3 Machine learning models

Machine learning, a powerful subset of artificial intelligence, has revolutionized numerous sectors, including financial markets. Its capabilities to learn from data, identify patterns, and make decisions with minimal human intervention have prompted researchers and practitioners alike to explore its potential for stock market prediction. The literature abounds with various machine learning techniques applied for this purpose, each boasting unique strengths and addressing different aspects of prediction challenges [69] [9] [81].

Supervised learning techniques, such as Support Vector Machines (SVM) and Decision Trees, have been proven to effectively predict stock market prices and trends based on historical data. Support Vector Machine (SVM) is an instrumental machine learning technique for data classification, aiming to identify a separation hyperplane that maximizes the margin between classes [18]. A substantial study using SVM was conducted by Fan et al. (2001), who sought to develop an efficient system to gain high profits by analyzing stock markets [26]. They employed SVM to select stocks that exceed the market's percentage return. Their method yielded a total return of 208% over 5 years, underscoring SVM's effectiveness [75]. Cao et al. (2003) were motivated by SVM's consistent results in forecasting financial time series [14]. They conducted a comparison of SVM, a retro-propagation multilayer neural network (BP), and a regularized radial base function (RBF) neural network using five datasets listed in the Chicago Mercantile Market. Results indicated SVM's superior performance in adaptive and free parameters [75].

Ballings et al. (2015) benchmarked ensemble methods comprising Random Forest, AdaBoost, and Kernel Factory against single classifier models, using data from 5767 publicly listed European companies. The performance measure was the Area Under the Curve (AUC) for long-term stock price direction prediction, with Random Forest emerging as the top algorithm [9]. Milosevic (2016) developed a classification approach for long-term prediction of stock market prices, where a stock is categorized as 'good' if its price increases by 10% in a year, otherwise 'bad' [57]. A manual feature selection was conducted, with Random Forest achieving the best F-Score of 0.751 against SVM and Naïve Bayes techniques [81].

Recent attention has also been drawn to the eXtreme Gradient Boosting (XGBoost) technique. Dey et al. (2016) used XGBoost algorithm to predict the direction of stocks, with technical indicators as features. The results showed that XGBoost outperformed other techniques, achieving an accuracy of 87-99% for long-term prediction of Apple and Yahoo stocks [23]. Zhang et al. (2018) proposed a stock price trend prediction system using a random forest model trained on historical data from the Shenzhen Growth Enterprise Market to classify stocks according to their close prices. This system proved robust against market volatility and outperformed several existing prediction methods [102]. Lv et al. (2019) evaluated various machine learning algorithms and DNN models, observing traditional machine learning algorithms performing better on directional evaluation indicators without transaction costs, while DNN models performed better considering transaction costs [51] [81].

Unsupervised learning methods have been instrumental in identifying correlations within complex, uncorrelated datasets such as the stock market. Powell et al. (2008) compared SVM and K-means after performing Principal Component Analysis (PCA) to reduce dimensions or features on S&P 500 data. The results indicated similar performances, with SVM achieving 89.1% and K-means reaching 85.6% [71] [81].

3.4 Deep learning models

Deep learning, a specialized subset of machine learning, leverages artificial neural networks with multiple hidden layers to make sense of high-dimensional and complex data structures. Complex market dynamics, the volume of data available, and the non-linear relationships between financial variables make deep learning particularly well suited to financial market prediction tasks.

Long Short-Term Memory (LSTM) networks, a form of Recurrent Neural Networks (RNNs), are the most popular and are well-suited to problems involving time-series data due to their ability to capture and retain patterns over time. An instance of this is the study by Chen et al. (2015), which utilized LSTM to model and predict returns of Chinese shares, and found that the LSTM model outperformed the random forecasting method [17]. LSTM was further employed by Fischer et al. (2017) for large-scale financial market predictions. Their experiment, which used data from the S&P 500 spanning from 1992 to 2015, revealed that LSTM performed better without taking advantage of classification methods such as random forest or deep neural network (DNN) [27] [75]. Lakshminarayanan et al. (2019), combined an LSTM model with crude oil price, gold price, and moving average to outperform both an LSTM model without these features and an SVM model [46]. Baek et al. (2018) proposed the ModAugNet framework for stock market index forecasting, which consisted of an overfitting prevention LSTM module and a prediction LSTM module [7]. Pang et al. (2020), proposed two improved deep LSTM models with embedded layers, showing improvement over the benchmark [66]. Moghar et al. (2020) aims to build a Long-Short Term Memory model (LSTM) to predict future stock market values and the main objective of this paper is to see in which precision it can predict and how much the epochs can improve our model [59]. A range of studies have been conducted that employ LSTM based models but with input parameters derived from a variety of sources. These sources include raw price data, technical and fundamental analysis, macroeconomic data, financial statements, news, and investor sentiment and many more [80].

Other forms of RNNs, such as the simple recurrent neural network (SRNN), the gated recurrent unit (GRU), and the LSTM were explored by Samarawickrama et al. (2017) for predicting future prices of selected companies listed on the Colombo Stock Exchange. The SRNN and LSTM networks were found to have superior accuracy in comparison to feedforward networks (FFNN). However, it was noted that the GRU networks did not perform as well for stock price forecasting [78] [75].

However, LSTM isn't the only deep learning model employed for financial forecasting. For example, the bat-neural network multi-agent system (BNNMAS) was proposed by Hafezi et al. (2015), as a method to predict the Deutscher Aktienindex (DAX) stock indices over eight years [34] [75].

The comparison between LSTM, a deep learning architecture, and traditional ARIMA model was performed by Siami Namini et al (2018). The aim of this comparison was to investigate if machine learning based approaches can outperform traditional methods in terms of accuracy. The results revealed that LSTM provided better overall performance than the ARIMA model [85] [75].

In the context of financial markets, FFNNs can be used for tasks such as predicting future prices of financial instruments based on past data, but are not that common. The use of Convolutional Neural Networks (CNNs) in stock price prediction is another key focus of deep learning research. CNNs, primarily designed for image processing, have demonstrated applicability to financial time-series data. Different researchers have ap-

plied them to stock market prediction using varying inputs and techniques [38]. Maqsood et al. (2020), developed a CNN model that used both price and sentiment analysis as input. He compared this model with linear regression and Support Vector Machines (SVM) and found that not all significant events have a significant impact on stock exchange prediction, emphasizing the influence of local events [55]. Sim et al. (2019), implemented a CNN network that employed 9 technical indicators but found no significant improvement in the stock market prediction performance from these indicators [86].

3.5 Hybrid models

Hybrid models in financial forecasting involve a combination of different statistical, machine learning, and sometimes even fuzzy logic approaches to achieve better results. The primary advantage of hybrid models lies in their ability to leverage the strengths of different models to capture patterns and trends that could not be recognized by individual models [81].

An example of a hybrid model is Markowska-Kaczmar and Dziedzic's (2008) supervised feedforward neural network, combined with the Pattern-Inflection Points (PIP) technique for dimensionality reduction, to identify patterns in stock data [56]. Another noteworthy example is Shen et al. (2012), that proposed a prediction algorithm that combined statistical methods with Support Vector Machines (SVMs). The technique leveraged correlations among global markets and commodities to predict future stock price trends, achieving prediction accuracy of 77.6% on the DJIA and up to 85% for longer-term predictions [83]. Wang et al. (2012) developed a Proposed Hybrid Model (PHM) that combined Exponential Smoothing Method (ESM), Autoregressive Integrated Moving Average (ARIMA), and a Backpropagation Neural Network (BPNN) model. The results indicated that the hybrid model outperformed each of the constituent sub-models and all traditional models when tested on the Shenzhen Integrated Index and DJIA [95]. Hybrid models have started to incorporate news events into the prediction process. For example, Yoshihara et al. (2014) combined Recurrent Neural Networks Restricted Boltzmann Machine (RNN-RBM) and Deep Belief Network (DBN) to predict stock trends based on the long-term effects of news events [100] [81].

Another interesting hybrid method was proposed, combining ARIMA with ANN models for financial time-series prediction, outperforming individual ARIMA and ANN models [6]. Similarly, Panigrahi et al. (2017) developed a high-efficiency methodology combining Exponential Smoothing State Space Model (ETS) and ANN models, outperforming methods like ARIMA, ETS, and Multilayer Perceptron (MLP) [67].A hybrid system combining long short-term memory (LSTM) and GARCH-type models to forecast stock price volatility was also introduced , delivering consistent results when tested on KOSPI 200 index data [45] [75]. Zhang et al. (2020) combines LSTM with Autoencoder and CNN for improved predictive results across financial and ML metrics [101] [64]. Wang et al. (2021), proposed a hybrid CNN-TLSTM (tanh-LSTM) model, effectively predicting the stock rate for complex nonlinear data, demonstrating improved performance over conventional MLP, CNN, and LSTM models, particularly evident in the USD/CNY exchange rate forecast. The model, composed of a convolutional layer for feature extraction from input data and a TLSTM component for time series analysis [94] [82].

Overall, these studies and methods underscore the increasing complexity and performance of hybrid models in finance, combining multiple statistical, machine learning, and even fuzzy logic methodologies to enhance prediction accuracy.

3.6 Other approaches

Beyond conventional machine learning models, various innovative methodologies have started making headway in financial forecasting, including reinforcement learning, sentiment analysis, and even quantum finance. These techniques bring unique perspectives to the complex challenge of predicting financial trends and activities [75].

Reinforcement learning (RL) is a dynamic machine learning approach where an agent learns to make decisions by interacting with an environment and receiving rewards or penalties for actions. Li proposed three distinct reinforcement learning methods, finding that Deep-Q Network (DQN) outperformed Double DQN and Duelling DQN [49]. Shin combined reinforcement learning with LSTM and CNN, the latter serving to extract features from generated charts of stock trading data, with the LSTM layer processing the features before RL defined the agent's actions [84]. Wu introduced an RL model with an LSTM-based agent to sense stock market dynamics and reduce manual indicator design difficulties [98], while Carapuço created an RL-Q network model to train RL agents in a novel simulated market environment [15] [38].

Sentiment analysis involves examining the sentiments or emotions expressed in written language. Bollen et al. (2019) investigated correlations between Twitter data and changes in the DJIA [12]. However, these results should be interpreted carefully, as not all Twitter users invest in stocks. Others focused on more direct sources of financial data, such as Lee et al. (2014)'s text analysis of company's 8-K reports, which reportedly improved prediction accuracy by 10% [47], and Kalyanaraman et al. (2016)'s sentiment polarity analysis of news articles [44] [81]. Das et. al. (2018) implemented sentiment analysis on Twitter posts along with the stock data for price forecasting using RNN [20]. Similarly, other authors used sentiment classification (neutral, positive, negative) for the stock open or close price prediction with various LSTM models [48]. They compared their results with SVM and achieved higher overall performance [80].

A growing interest in intelligent financial systems has led to the development of quantum finance theory, which offers dynamic arbitrage possibilities and potential solutions for trading optimization, risk profiling, and prediction. Baaquie's work is particularly notable, leveraging quantum finance theory to price rate range accrual swaps [5]. Orus et al. (2019), discusses how quantum computation can be applied to financial problems, providing an overview of current approaches and potential prospects [65]. Paquet et al. (2022), introduces a new hybrid deep quantum neural network for financial predictions, and demonstrates the accuracy and efficiency of the system [68]. Other researchers proposed systems based on fuzzy logic, genetic algorithms, and chaotic theory to enhance financial predictions [75].

The potential to harness the computational power of quantum computers, the dynamic learning of RL, and the real-time emotional pulse-check of sentiment analysis could revolutionize our approach to financial forecasting in the years to come.

4. APPROACH

In this chapter, the research design and methodologies utilized are presented for predicting the closing price of the S&P 500 index. The regression approach is chosen to forecast the exact closing price of the S&P 500 index, rather than a classification approach which would only predict the market's directional movement. This choice is predicated on the desire to deliver a more in-depth perspective into the magnitude of future index values.

The current implementation involves 12 different prediction models for the S&P 500, encompassing Statistical (ARIMA), Machine Learning (SVR), and Deep Learning (LSTM) methodologies, each applied to four types of data based on different forecasting horizons and sets of features. The approach comprises four main stages: data collection and preprocessing, modeling, evaluation, and trading simulation. Each stage and its respective components are thoroughly explained. An illustration of these stages is provided in Figure 4.1.

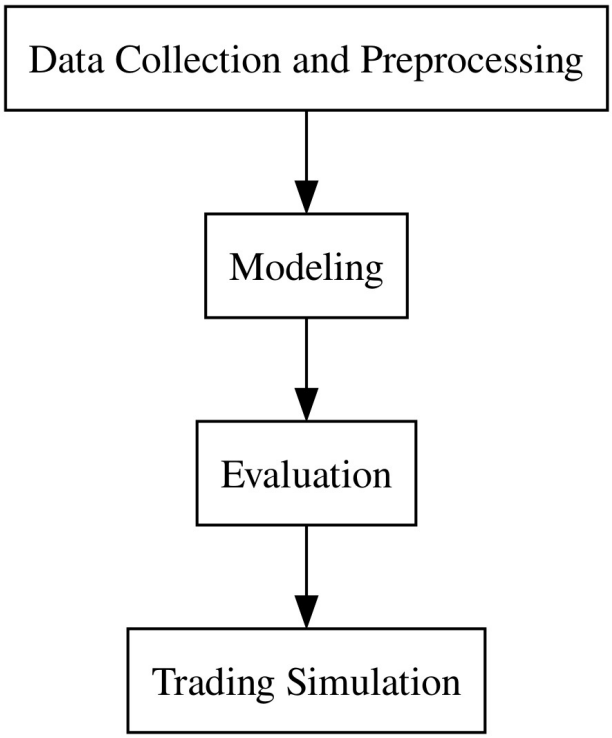

Figure 4.1: Approach Stages

4.1 Workflow and goals

The workflow follows a structured process that starts with data collection and preprocessing. This step involves gathering data from reliable sources and refining it to create a dataset appropriate for the predictive models. Modeling is the next step, and with the data prepared, the different types of models are implemented, tailored to the specific data characteristics. After the models are trained and tested, their performance and accuracy is evaluated using different metrics. This provides an objective assessment of how well each model has learned from the data and can predict. Finally the models' predictions are used in a practical trading simulation. This step helps understand how the models' predic-

tions would perform in a real-world trading environment, thereby providing an additional layer of validation.

The primary goal of this research is to gain a deeper understanding of financial market dynamics and the potential of various predictive methodologies in forecasting the closing price of a major stock index. The workflow is structured to not only explore different prediction models, but also to practically apply these findings via a trading simulation.

This study also serves as a means, to dig deeper into the combination of technology and finance, enhancing the understanding of the FinTech field. By implementing and evaluating these methodologies firsthand, the aim is to further grasp the complexities of the stock market and refine his skills in predictive modeling and algorithmic trading.

4.2 Why S&P 500?

The choice of the S&P 500 as the subject of this research is primarily motivated by its representative and influential status in the financial world. The S&P 500, or Standard & Poor's 500, is a stock market index that gauges the performance of 500 large companies listed on stock exchanges in the United States. It's one of the most widely followed equity indices and is broadly considered as a bellwether of the U.S. stock market's performance.

Several factors justify the selection of the S&P 500 over a single company's stock or other indices. Unlike individual company stocks, which can be heavily influenced by company-specific news and events, the S&P 500 offers a more comprehensive reflection of the market's overall state, encompassing a diverse range of sectors. This not only provides a more stable and consistent data set for prediction purposes, but also makes the findings potentially more broadly applicable.

Ultimately, the goal of forecasting the S&P 500 is to predict its specific future performance, to gain insights into the broader market trends and explore the relative strengths of various predictive methodologies in financial forecasting.

4.3 Data collection and preprocessing

The focus is on the initial stages of the research process: data collection and preprocessing. These preliminary steps lay the groundwork for all the subsequent stages, ensuring the data used for modeling is accurate, relevant, and prepared in a way that optimizes the predictive algorithms' performance. The tools employed in these steps are detailed, followed by the description of the data collection process, data processing, and feature selection.

4.3.1 Tools used

The Python programming language was used for all stages of data handling and modeling. The code was developed and executed on Google Colaboratory, a cloud-based Python notebook environment. This choice allowed for seamless integration with data sources, easy manipulation and analysis of data, and efficient implementation of various models.

A wide range of Python libraries were employed throughout this research, each playing a crucial role in different stages of the process. The following table (Table 4.1) provides a

brief description of each library and its function in the study.

Table 4.1: Libraries

Library	Description
copy	Provides functions that allow duplicate copies of existing objects
datetime	To manipulate dates and times
dateutil.relativedelta	Provides capabilities for performing arithmetic with dates and times
fredapi	To access a vast amount of economic data from the Federal Reserve Bank of St. Louis's database
IPython.display	For pretty printing and display tools
json	For parsing and manipulating JSON data
keras	High-level neural networks API, for constructing and training deep learning models
matplotlib.pyplot	For creating static, animated, and interactive visualizations in Python
numpy	For numerical computing with powerful N-dimensional array object
pandas	For data manipulation and analysis
pmdarima.arima	For automatic ARIMA modeling
pytz	For manipulating and formatting dates and times in Python
pywt	For wavelet transformation functions
random	Generates pseudo-random numbers
seaborn	Data visualization library based on matplotlib
sklearn	Machine learning library with various classification, regression and clustering algorithms
statsmodels.robust.mad	Provides Median Absolute Deviation functions
statsmodels.tsa.arima.model	For ARIMA models
talib	Technical analysis library for financial trading applications
tensorflow	An end-to-end open source platform for machine learning
textwrap	Provides functions for wrapping input paragraph text and filling strings
tqdm	Progress bar utility for visual feedback
yfinance	Used to download historical market data from Yahoo finance

4.3.2 Data collection

The data collection process forms the basis of this study. Reliable and pertinent data are integral to the accuracy and relevance of the predictive models built.

4.3.2.1 Data sources

The data were gathered from two primary sources, ”yfinance” and the ”Federal Reserve Economic Data (FRED)”. Additionally some technical indicators were calculated using ”The Technical Analysis Library (TA-Lib)”.

The Python library called ”yfinance” is what allows users to download historical market data from Yahoo Finance. It enables users to access stock information, including closing prices, open prices, high prices, low prices, volumes, split history, and more, directly into Python. You can also pull in information about company dividends, stock splits, and other relevant financial details. It is widely used by researchers and investors for financial analysis and algorithmic trading models.

”FRED API (Federal Reserve Economic Data API)” is a service provided by the Federal Reserve Bank of St. Louis that gives users access to a vast amount of U.S. and international economic data. It includes data about employment, GDP, interest rates, exchange rates, consumer price indexes, and more. Users can access and retrieve this data programmatically using the FRED API, which makes it useful for economic research, forecasting, and policy-making.

The ”Technical Analysis Library (TA-Lib)” is another Python library widely used among trading and investment communities. It is an open-source software that provides tools for technical analysis of financial markets. TA-Lib includes over 150 functions for various types of analysis, including pattern recognition, moving averages, oscillators, and more. It can be used to build algorithmic trading strategies and conduct financial market analysis.

4.3.2.2 Scenarios

This data is divided into four different scenarios, each represented by a different dataset:

- Price history predicting 1-day ahead closing prices.
- Price history predicting 30-days ahead closing prices.
- Multiple features predicting 1-day ahead closing prices.
- Multiple features predicting 30-days ahead closing prices.

The structure and features of the datasets are detailed in the following tables (Table 4.2, 4.3, 4.4 and 4.5).

The ”Price History” dataset (Table 4.2) solely focuses on the historical price data of the S&P 500 index, including opening, high, low, and closing prices. It is used to predict the closing price of the S&P 500 index either on the next trading day or 30 days ahead.

On the other hand, the ”Multiple Features” datasets is represented by three tables, each corresponding to a different data source.

Table 4.3 describes the features obtained from yfinance. These include the volume, opening, high, low, and closing prices of the S&P 500 index on a given trading day, as well as the closing prices of Philadelphia Gold and Silver Index (XAU), the exchange rate between the US Dollar and the British Pound (USD/GBP), the closing price of a barrel of crude oil in the US, and the exchange rate between the Chinese Yuan and the US Dollar (CNY/USD).

Table 4.2: Description of the Features Used in the ”Price History” Datasets

Feature	Source	Description
Open	yfinance	Opening price of the S&P 500 index on a given trading day.
High	yfinance	Highest price of the S&P 500 index on a given trading day.
Low	yfinance	Lowest price of the S&P 500 index on a given trading day.
Close	yfinance	Closing price of the S&P 500 index on a given trading day.
Next_Day_Close	yfinance	Closing price of the S&P 500 index on the next trading day.
Next_30_Day_Close	yfinance	Closing price of the S&P 500 index 30 days ahead.

Table 4.3: Description of the Features Obtained from yfinance Used in the ”Multiple Features” Datasets

Feature	Source	Description
Open	yfinance	Opening price of the S&P 500 index on a given trading day.
High	yfinance	Highest price of the S&P 500 index on a given trading day.
Low	yfinance	Lowest price of the S&P 500 index on a given trading day.
Close	yfinance	Closing price of the S&P 500 index on a given trading day.
Volume	yfinance	Number of shares traded during a given trading day.
Year, Month,Day, Day of week	yfinance	Date data
XAU_Close	yfinance	Closing price of Philadelphia Gold and Silver Index (XAU) on a given trading day.
USD/GBP Exchange Rate	yfinance	The exchange rate between the US Dollar and the British Pound on a given trading day.
Crude Oil US Prices	yfinance	The closing price of a barrel of crude oil in the US on a given trading day.
CNY/USD Exchange Rate	yfinance	The exchange rate between the Chinese Yuan and the US Dollar on a given trading day.
Next_Day_Close	yfinance	Closing price of the S&P 500 index on the next trading day.
Next_30_Day_Close	yfinance	Closing price of the S&P 500 index 30 days ahead.

This table also describes the features used to represent the closing price of the S&P 500 index on the next trading day and 30 days ahead. Additionally it includes some date data like Year, Month, Day and Day of the week.

Table 4.4 provides a detailed description of the calculated metrics, used in the dataset. These include volatility, Relative Strength Index (RSI), Moving Average Convergence Di-

Table 4.4: Description of the Calculated Features Used in the ”Multiple Features” Datasets

Feature	Source	Description
Volatility-10day	TA-Lib	10-day rolling standard deviation of daily returns.
Volatility-30day	TA-Lib	30-day rolling standard deviation of daily returns.
RSI	TA-Lib	Relative Strength Index, a momentum oscillator that measures speed and change of price movements.
MACD	TA-Lib	Moving Average Convergence Divergence, a trend-following momentum indicator.
MACD_signal	TA-Lib	Signal line for the MACD, calculated as the 9-day EMA of the MACD.
MACD_hist	TA-Lib	MACD histogram, calculated as the difference between MACD and its signal line.
BB_upper	TA-Lib	Upper Bollinger Band, calculated as SMA + 2 standard deviations.
BB_middle	TA-Lib	Middle Bollinger Band, calculated as the simple moving average (SMA).
BB_lower	TA-Lib	Lower Bollinger Band, calculated as SMA - 2 standard deviations.
SMA_5	TA-Lib	5-day Simple Moving Average of closing prices.
SMA_15	TA-Lib	15-day Simple Moving Average of closing prices.
SMA_30	TA-Lib	30-day Simple Moving Average of closing prices.
stochastic_K	TA-Lib	Stochastic oscillator %K, a momentum indicator.
stochastic_D	TA-Lib	Stochastic oscillator %D, a 3-day simple moving average of %K.

Table 4.5: Description of the Features Obtained from FRED Used in the ”Multiple Features” Datasets

Feature	Source	Description
Federal Funds Effective Rate	FRED	The interest rate at which depository institutions trade federal funds with each other overnight.
US GDP	FRED	The gross domestic product of the United States, a broad measure of economic activity.
US Unemployment Rate	FRED	The unemployment rate in the United States, a measure of labor market health.
US Consumer Price Index for All Urban Consumers: All Items Less Food and Energy	FRED	A measure of price changes in consumer goods and services, excluding food and energy.
US Consumer Price Index for All Urban Consumers: All Items	FRED	A measure of price changes in consumer goods and services.

vergence (MACD), Bollinger Bands, and Simple Moving Averages (SMA). It also includes the Stochastic Oscillator, a momentum indicator.

Finally, Table 4.5 details the macroeconomic indicators obtained from the Federal Reserve Economic Data (FRED). These indicators provide broader context on economic condi-

tions, including the Federal Funds Effective Rate, US GDP, US Unemployment Rate, and the US Consumer Price Index for All Urban Consumers, both with and without food and energy.

These tables provide a comprehensive view of the features used in this study, and what they represent.

4.3.2.3 Feature categories

The features used in the ”Multiple Features” dataset belong into five main categories: Market Features, Technical Indicators, Macro Economic Indicators, Date Data, and Target Variables. These categories are visualized in the following Table 4.6.

Table 4.6: Classification of the Features in the ”Multiple Features” Datasets

Indicator Group	Features
Market Features	Open, High, Low, Close, Volume, XAU_Close, USD/GBP Exchange Rate, Crude Oil US Prices, CNY/USD Exchange Rate
Technical Indicators	Volatility-10day, Volatility-30day, RSI, MACD, MACD_signal, MACD_hist, BB_upper, BB_middle, BB_lower, SMA_5, SMA_15, SMA_30, stochastic_K, stochastic_D
Macro Economic Indicators	Federal Funds Effective Rate, US GDP, US Unemployment Rate, US Consumer Price Index for All Urban Consumers: All Items Less Food and Energy, US Consumer Price Index for All Urban Consumers: All Items
Date Data	Year, Month, Day, Day_of_week
Target Variables	Next_Day_Close, Next_30_Day_Close

Market Features are direct measures of market activity. They include the opening, high, low, and closing prices of the S&P 500 index on a given trading day, the number of shares traded (Volume), the closing prices of Philadelphia Gold and Silver Index (XAU), exchange rates (USD/GBP and CNY/USD), and US crude oil prices.

Technical Indicators are derived from market features and are used extensively in technical analysis to predict future price movements. They include measures of volatility, momentum indicators such as the Relative Strength Index (RSI) and the Stochastic Oscillator, trend-following indicators like the Moving Average Convergence Divergence (MACD) and Simple Moving Averages (SMA), and volatility bands like the Bollinger Bands.

Macro Economic Indicators provide a broader context of the overall economic conditions. These include the Federal Funds Effective Rate, which is the interest rate at which depository institutions trade federal funds with each other overnight, the gross domestic product (GDP) of the US, which is a broad measure of economic activity, the unemployment rate, which is a measure of labor market health, and the Consumer Price Index for All Urban Consumers, a measure of price changes in consumer goods and services, both with and without food and energy.

Date Data are instrumental in capturing temporal patterns, seasonal trends, and cyclical behaviors, thereby enhancing the model’s predictive accuracy. Additionally, they facilitate

advanced feature engineering and time-series analysis, allowing for better understanding of data progression.

Target Variables are the features that the model aims to predict. In this dataset, these are the closing price of the S&P 500 index on the next trading day (Next_Day_Close) and the closing price 30 days ahead (Next_30_Day_Close).

4.3.2.4 Index visualization

Visualizing the data is a vital step in understanding patterns and trends present in the dataset. The data for this study spans three decades, from 1993 to 2023, capturing a wide range of market conditions and economic cycles.

Figures 4.2, 4.3, 4.4, and 4.5, illustrate snippets of the four different scenarios considered in this study: predicting the next day's closing price using only market features, predicting the closing price 30 days ahead using only market features, predicting the next day's closing price using multiple features, and predicting the closing price 30 days ahead using multiple features. Each scenario presents its unique challenges and insights, contributing to a comprehensive understanding of the predictability of the S&P 500 index.

Date	Open	High	Low	Close	Next_Day_Close
1993-06-01	450.23001099	455.63000488	450.23001099	453.82998657	453.85000610
1993-06-02	453.82998657	454.52999878	452.67999268	453.85000610	452.48999023
1993-06-03	453.83999634	453.85000610	451.11999512	452.48999023	450.05999756
1993-06-04	452.42999268	452.42999268	448.92001343	450.05999756	447.69000244
1993-06-07	450.07000732	450.75000000	447.32000732	447.69000244	444.70999146
...	...	...	...	...	...
2023-05-23	4176.79980469	4185.68017578	4142.54003906	4145.58007812	4115.24023438
2023-05-24	4132.95996094	4132.95996094	4103.97998047	4115.24023438	4151.27978516
2023-05-25	4155.70996094	4165.74023438	4129.72998047	4151.27978516	4205.45019531
2023-05-26	4156.16015625	4212.87011719	4156.16015625	4205.45019531	4205.52001953
2023-05-30	4226.70996094	4231.10009766	4192.18017578	4205.52001953	4179.83007812

7554 rows × 5 columns

Figure 4.2: S&P 500 Next Day's Closing Price Using Only Market Features

Figure 4.6 presents the direction of the closing price of the S&P 500 index from 1993 to 2023. Over this period, the index shows a general upward trend, reflecting the growth of the market and the economy. However, the path is not linear and is punctuated with periods of volatility and downturns, which underscores the complexity and dynamic nature of the stock market.

Date	Open	High	Low	Close	Next_30_Day_Close
1993-06-01	450.23001099	455.63000488	450.23001099	453.82998657	450.07998657
1993-06-02	453.82998657	454.52999878	452.67999268	453.85000610	449.22000122
1993-06-03	453.83999634	453.85000610	451.11999512	452.48999023	445.75000000
1993-06-04	452.42999268	452.42999268	448.92001343	450.05999756	446.02999878
1993-06-07	450.07000732	450.75000000	447.32000732	447.69000244	447.30999756
...	...	...	...	...	...
2023-04-12	4121.72021484	4134.37011719	4086.93994141	4091.94995117	4115.24023438
2023-04-13	4100.04003906	4150.25976562	4099.39990234	4146.22021484	4151.27978516
2023-04-14	4140.10986328	4163.18994141	4113.20019531	4137.64013672	4205.45019531
2023-04-17	4137.16992188	4151.72021484	4123.18017578	4151.31982422	4205.52001953
2023-04-18	4164.25976562	4169.47998047	4140.35986328	4154.87011719	4179.83007812

7525 rows × 5 columns

Figure 4.3: S&P 500 Closing Price 30 Days Ahead Using Only Market Features

Date	Open	High	Low	Close	Year	Month	Day	Day_of_week	Volume	Volatility-10day	Volatili 30
1993-06-01	450.23001099	455.63000488	450.23001099	453.82998657	1993	6	1	1	229690000	0.01143362	0.03011
1993-06-02	453.82998657	454.52999878	452.67999268	453.85000610	1993	6	2	2	295560000	0.01143362	0.03011
1993-06-03	453.83999634	453.85000610	451.11999512	452.48999023	1993	6	3	3	285570000	0.01143362	0.03011
1993-06-04	452.42999268	452.42999268	448.92001343	450.05999756	1993	6	4	4	226440000	0.01143362	0.03011
1993-06-07	450.07000732	450.75000000	447.32000732	447.69000244	1993	6	7	0	236920000	0.01143362	0.03011
...	...	...	...	...	...	...	...	...	...	...	
2023-05-23	4176.79980469	4185.68017578	4142.54003906	4145.58007812	2023	5	23	1	4155320000	0.02183779	0.04537
2023-05-24	4132.95996094	4132.95996094	4103.97998047	4115.24023438	2023	5	24	2	3739160000	0.02271376	0.04580
2023-05-25	4155.70996094	4165.74023438	4129.72998047	4151.27978516	2023	5	25	3	4147760000	0.02445115	0.04468
2023-05-26	4156.16015625	4212.87011719	4156.16015625	4205.45019531	2023	5	26	4	3715460000	0.02726502	0.04645
2023-05-30	4226.70996094	4231.10009766	4192.18017578	4205.52001953	2023	5	30	1	4228510000	0.02730751	0.04637

7554 rows × 34 columns

Figure 4.4: S&P 500 Next Day's Closing Price Using Multiple Features

Date	Open	High	Low	Close	Year	Month	Day	Day_of_week	Volume	Volatility-10day	Volatilit 30d
1993-06-01	450.23001099	455.63000488	450.23001099	453.82998657	1993	6	1	1	229690000	0.01143362	0.030114
1993-06-02	453.82998657	454.52999878	452.67999268	453.85000610	1993	6	2	2	295560000	0.01143362	0.030114
1993-06-03	453.83999634	453.85000610	451.11999512	452.48999023	1993	6	3	3	285570000	0.01143362	0.030114
1993-06-04	452.42999268	452.42999268	448.92001343	450.05999756	1993	6	4	4	226440000	0.01143362	0.030114
1993-06-07	450.07000732	450.75000000	447.32000732	447.69000244	1993	6	7	0	236920000	0.01143362	0.030114
...	...	...	...	...	...	...	...	...	...	...	
2023-04-12	4121.72021484	4134.37011719	4086.93994141	4091.94995117	2023	4	12	2	3633120000	0.02203501	0.055152
2023-04-13	4100.04003906	4150.25976562	4099.39990234	4146.22021484	2023	4	13	3	3596590000	0.02149877	0.056126
2023-04-14	4140.10986328	4163.18994141	4113.20019531	4137.64013672	2023	4	14	4	3575690000	0.02178261	0.055901
2023-04-17	4137.16992188	4151.72021484	4123.18017578	4151.31982422	2023	4	17	0	3611180000	0.01715570	0.053817
2023-04-18	4164.25976562	4169.47998047	4140.35986328	4154.87011719	2023	4	18	1	3536640000	0.01689806	0.053816

7525 rows × 34 columns

Figure 4.5: S&P 500 Closing Price 30 Days Ahead Using Multiple Features

Figure 4.6: S&P 500 Direction

4.3.3 Preprocessing

Data preprocessing involved several steps to ensure the data is suitable for building predictive models.

4.3.3.1 Handling missing data

The data covers a period from 1993 to the end of 2023. One of the unique challenges of financial data is the occurrence of missing data due to weekends and holidays when the stock markets are closed. Care was taken to handle these missing values in a manner that preserves the integrity of the data and prevents future leakage.

A check is performed to identify any missing values (NaNs) in the data. This is done for each column in the dataset. After identifying the number of missing values in each column, a two-step process is used to fill these missing values. The primary method used is forward filling, where each missing value is replaced with the preceding value. This method is chosen to avoid leakage of future data. However, if any missing values remain after forward filling, backward filling is used as a secondary method to fill these missing values. This is particularly useful for filling missing values at the start of the dataset, where no preceding values are available.

Notably, the data is not uniformly daily. Some features, such as the Federal Funds Effective Rate and macroeconomic indicators like GDP, Unemployment Rate, and Consumer Price Index, are available at a monthly or quarterly frequency. For these features, the data is joined to the main dataset on the index, effectively propagating the last known value to fill the gaps until a new value is available. This approach of forward filling is reasonable when dealing with macroeconomic indicators that change relatively slowly.

Initially, an approach was considered to make the whole dataset business days by filling holidays. However, this approach was abandoned due to the degradation in performance it caused. Filling the holidays led to an over-representation of non-operational days, which introduced noise and distorted the trends and patterns in the data.

4.3.3.2 Creating target variables

The target variables, the closing price of the S&P 500 index on the next trading day and 30 days ahead, were created by shifting the closing price, into a new column. It is important to note that the 30-day ahead predictions refer to 30 market days, not calendar days. Market days are business days excluding holidays, thus reflecting the actual operating days of the stock market. This aligns with the reality of stock market operations, enhancing the practical relevance of the predictions.

4.3.3.3 Data split

Splitting data into training, validation, and test sets is a crucial part of the data preprocessing stage in machine learning workflows. This is necessary to properly train and evaluate the performance of the model. The training set is used to train the model, the validation set is used to fine-tune the model parameters and select the best model, and the test set is used to provide an unbiased evaluation of the final model fit.

In this research, the data is split chronologically, with the first 80% used for training and the remaining 20% used for testing. The split ensures that the models are trained on past data and tested on future data, mimicking a real-world scenario where the goal is to predict future stock prices based on past data.

To visualize the data split (Figure 4.7), line plots are created for each dataset, where the

x-axis represents time and the y-axis represents the closing price. The training data is plotted in one color, and the testing data is plotted in another color.

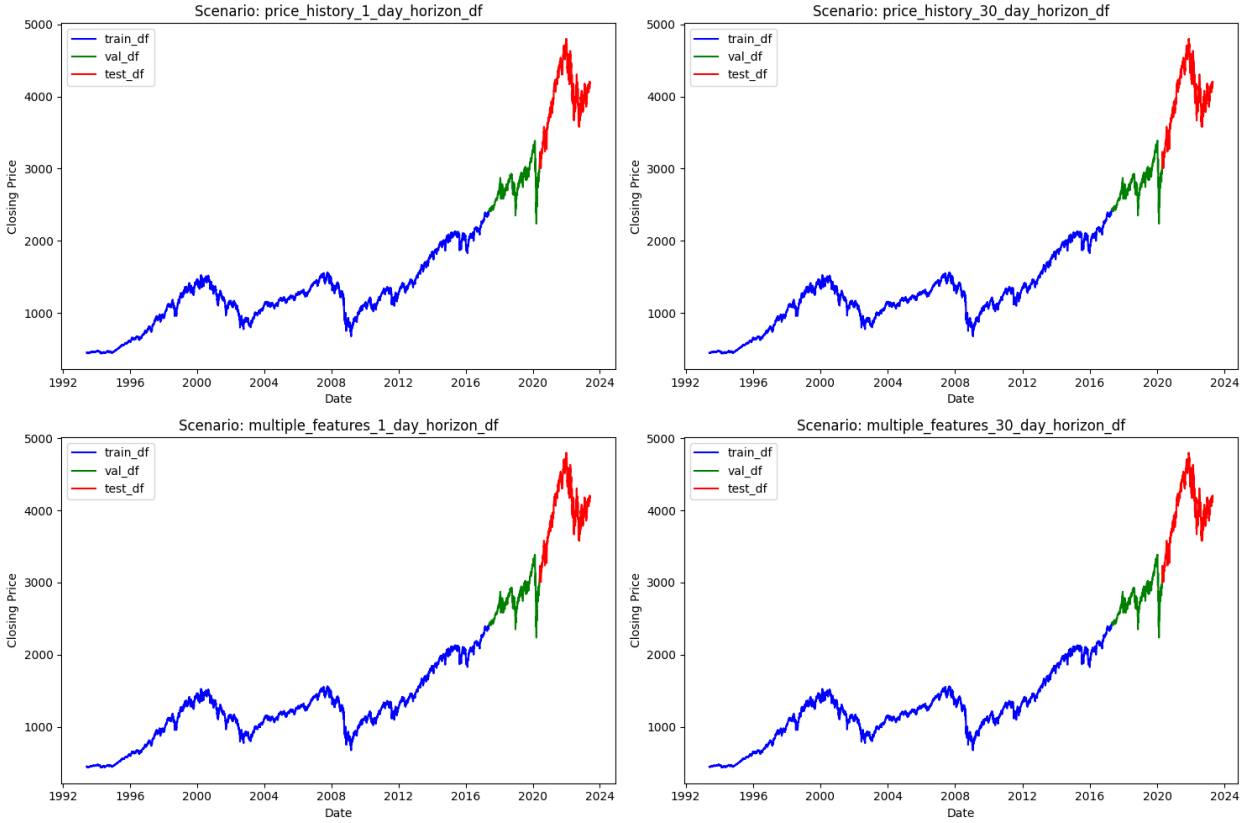

Figure 4.7: Data Split Plots

The four scenarios that were considered, each has three distinct datasets: training, validation, and testing. Each scenario and corresponding dataset is summarized in Table 4.7. The 'Price history predicting 1-day ahead closing prices' and 'Price history predicting 30-days ahead closing prices' scenarios use only market data with five features. The 'Multiple features predicting 1-day ahead closing prices' and 'Multiple features predicting 30-days ahead closing prices' scenarios include multiple features amounting to 34 in total. The date range of the data varies between datasets due to the different prediction horizons. The training datasets contain the bulk of the data, ranging from 1993 to 2017. The validation and test datasets cover more recent years, from 2017 to 2023, ensuring the models are evaluated on unseen, out-of-sample data. This separation of data is crucial for assessing the predictive performance of the models.

4.3.3.4 Feature correlation

Feature correlation is another important aspect of preprocessing. It involves determining the relationship between different features in the dataset. Features that are highly correlated can lead to redundancy in the model, which might reduce the model's performance. Feature correlation is performed on the training set, to ensure objectivity and no future leakage.

The correlation of each feature with the target variable, and each other is calculated. The correlation values range between -1 and +1, where a value close to +1 indicates a strong positive correlation, a value close to -1 indicates a strong negative correlation, and a value

Table 4.7: Dataset Information for Each Scenario

Scenario	Dataset	Shape	First Index	Last Index
Price history predicting 1-day ahead closing prices	train_df	(6043, 5)	1993-06-01	2017-05-26
	val_df	(755, 5)	2017-05-30	2020-05-28
	test_df	(756, 5)	2020-05-29	2023-05-30
Price history predicting 30-days ahead closing prices	train_df	(6020, 5)	1993-06-01	2017-04-25
	val_df	(752, 5)	2017-04-26	2020-04-21
	test_df	(753, 5)	2020-04-22	2023-04-18
Multiple features predicting 1-day ahead closing prices	train_df	(6043, 34)	1993-06-01	2017-05-26
	val_df	(755, 34)	2017-05-30	2020-05-28
	test_df	(756, 34)	2020-05-29	2023-05-30
Multiple features predicting 30-days ahead closing prices	train_df	(6020, 34)	1993-06-01	2017-04-25
	val_df	(752, 34)	2017-04-26	2020-04-21
	test_df	(753, 34)	2020-04-22	2023-04-18

around 0 suggests no correlation. This analysis is visualized using a correlation matrix heatmap. Each cell in the heatmap represents the correlation coefficient between two features, and the color of each cell reflects the magnitude of this coefficient.

Correlation heatmaps can help identify multicollinearity, which is when two or more features are highly correlated. Multicollinearity can pose a problem in some models, as it can make the model's estimates unstable and harder to interpret. Features that have strong correlation (either positive or negative) with the target variable could be good predictors for the model. On Figures 4.8 and 4.9, one of the 4 correlation heatmaps is presented as an example.

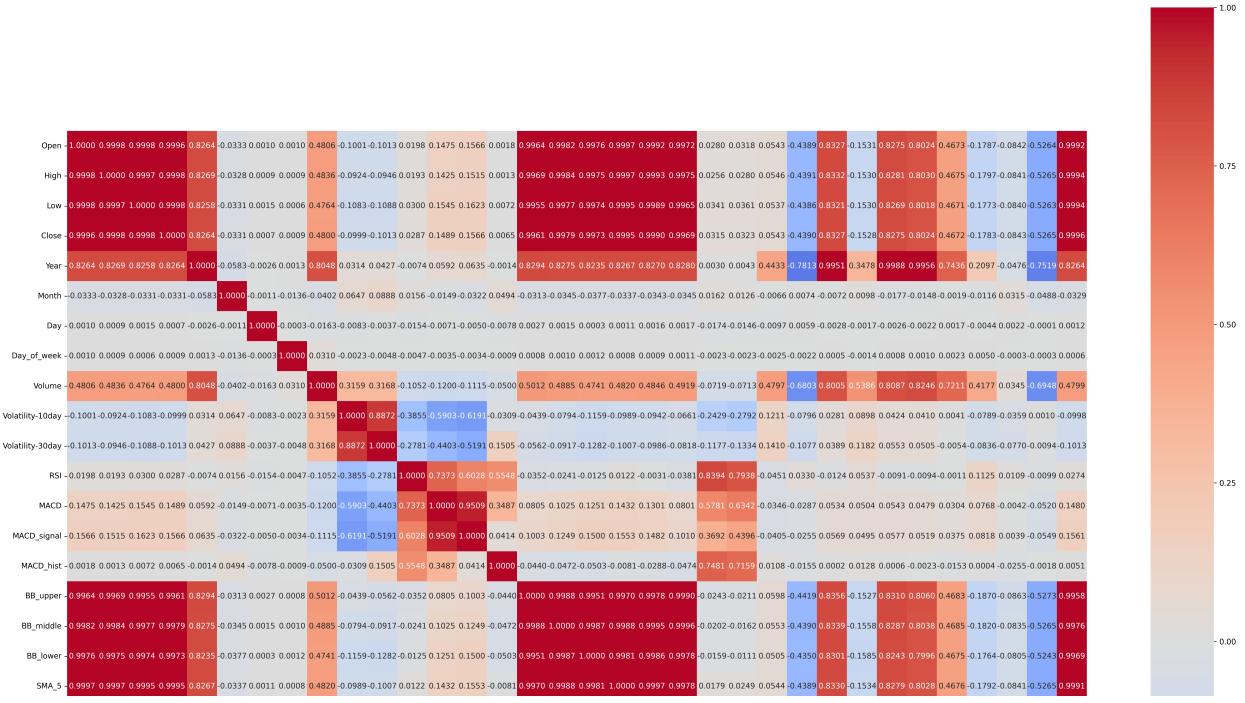

Figure 4.8: Correlation Matrix Heatmap (Part 1)

Next, a Random Forest Regressor is trained on the training data, and feature importance is extracted from the model. This gives an insight into how much each feature contributes to the model's predictive power. To select the most important features, a threshold is set, and any features with importance over the threshold are shown.

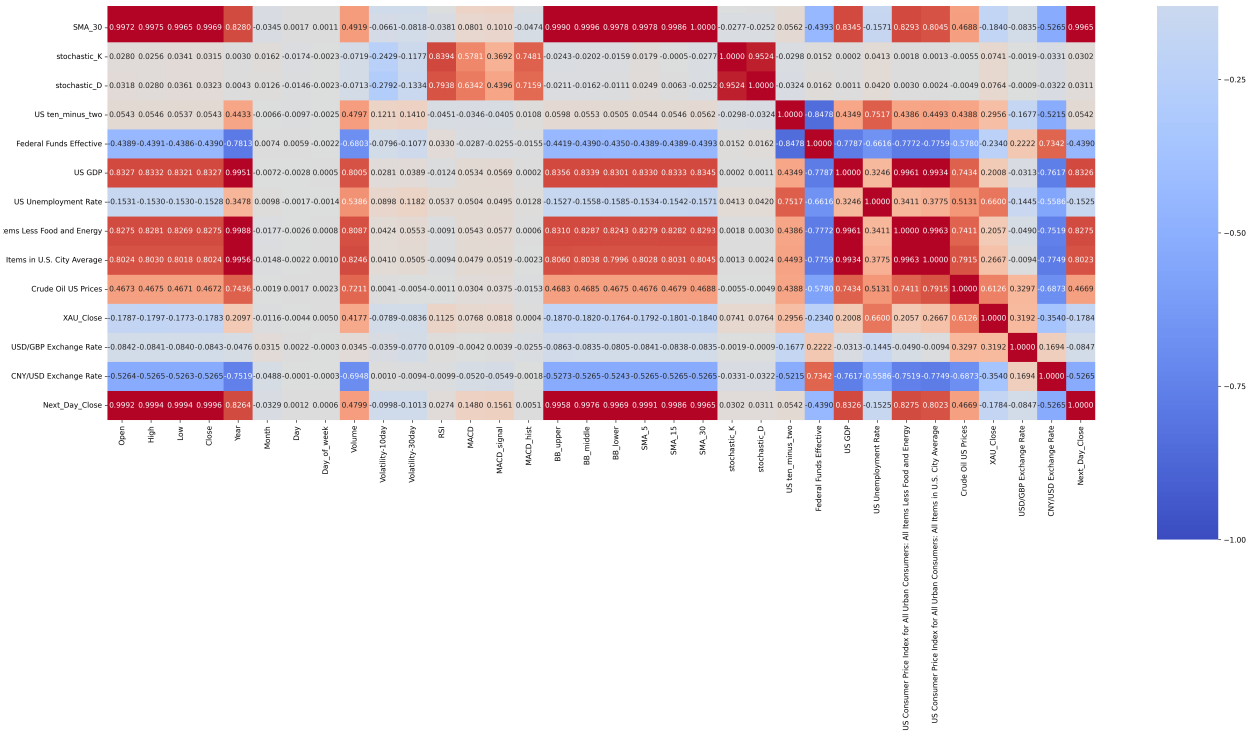

Figure 4.9: Correlation Matrix Heatmap (Part 2)

Although feature correlation analysis revealed high collinearity among some variables and negligible correlation between others and the target variable, it was decided to retain all the features within the dataset. This decision was made after model experimentation, which indicated that models trained on selected feature subsets, did not significantly improve predictive performance over models trained on the entire feature set. In fact, some results even showed worse performance. As a result, in the interest of maintaining the richness of the data, even features perceived as 'neutral' or 'low importance' or 'very related' were included, under the consideration that they might reveal relationships in the data, and contribute to the model.

4.3.3.5 Denoising

A key stage of the preprocessing workflow includes denoising the dataset, which aims to reduce noise within the data while maintaining the primary trend. This technique is applied across all features within each dataset, using the wavelet transform as the denoising method. Wavelet transform, a powerful mathematical function, divides the data into distinct frequency components and analyzes each component with resolution suitable to its scale. This decomposition allows us to inspect individual frequency subbands in isolation.

The crucial parameters directing the degree of denoising include the threshold value, the wavelet type, and the decomposition level. The threshold value identifies which coefficients constitute noise, while the wavelet type can capture various features in the data. Furthermore, the decomposition level can be adjusted to control the amount of denoising. The denoising threshold, selected based on the universal threshold formula, helps to determine which coefficients should be treated as noise, assuming white Gaussian noise.

Different wavelets such as Daubechies 2 (db2), Daubechies 8 (db8), and Haar ('haar') can be used depending on the nature of the data. Each of these functions captures different features in the data. After experimenting with various wavelet functions and decomposition

levels, the Haar wavelet with a decomposition level of 7 was selected, as it provided the best balance between denoising and preserving the original signal's structure.

While denoising the data, the primary focus is on the training dataset, manipulating various parameters and examining the effect on the validation set. The process was applied first to a single target column, and the results were reviewed before proceeding to denoise all the dataset columns that made sense denoising (without Date, month, year, etc.). Better performance was noted when denoising all these dataset columns. The Figure 4.10 shows snippets of the original dataset versus the denoised dataset of one scenario, and Figure 4.11 shows the difference between original and denoised dataset's target column.

4.3.3.6 Feature scaling

Feature scaling is an important step in many machine learning algorithms. It ensures that all features have a similar scale, which can help the algorithm converge more quickly and improve performance. In this study, three types of feature scaling were applied into the training set and experimented with: Standard Scaler, MinMax Normalizer, and no scaling.

- **Standard Scaler** standardizes features by removing the mean and scaling to unit variance.
- **MinMax Normalizer** scales and translates each feature individually such that it is in the given range on the training set, typically between zero and one.
- **No scaling** means the raw data was used without any modification to the scale.

Through experimentation, it was found that the Standard Scaler worked best with this data. This could be because many algorithms assume that all features are centered around zero and have approximately the same variance. When features have different ranges, the ones with larger ranges may dominate the ones with smaller ranges. Standard Scaler helps to prevent this by ensuring all features have the same scale.

4.4 Modeling and evaluation

In this chapter, the modeling process is presented, where three different methodologies are employed: Statistical (ARIMA), Machine Learning (SVR), and Deep Learning (LSTM). These models are applied to the four types of data, each characterized by distinct forecasting horizons (1 or 30) and sets of features (price history or multiple).

4.4.1 Statistical modeling (ARIMA)

The first method that is used for the prediction model is the ARIMA model, an acronym that stands for AutoRegressive Integrated Moving Average. ARIMA is a forecasting technique that projects the future values of a series, based on its own historical patterns.

In this implementation, the ARIMA model is configured with the parameters (p, d, q). These parameters are defined as follows:

train_df

Date	Open	High	Low	Close	Next_Day_Close
1993-06-01	450.23001099	455.63000488	450.23001099	453.82998657	453.85000610
1993-06-02	453.82998657	454.52999878	452.67999268	453.85000610	452.48999023
1993-06-03	453.83999634	453.85000610	451.11999512	452.48999023	450.05999756
1993-06-04	452.42999268	452.42999268	448.92001343	450.05999756	447.69000244
1993-06-07	450.07000732	450.75000000	447.32000732	447.69000244	444.70999146
...	...	...	...	...	...
2017-05-22	2387.20996094	2395.45996094	2386.91992188	2394.02001953	2398.41992188
2017-05-23	2397.04003906	2400.85009766	2393.87988281	2398.41992188	2404.38989258
2017-05-24	2401.40991211	2405.58007812	2397.98999023	2404.38989258	2415.07006836
2017-05-25	2409.54003906	2418.70996094	2408.01000977	2415.07006836	2415.82006836
2017-05-26	2414.50000000	2416.67993164	2412.19995117	2415.82006836	2412.90991211

6043 rows × 5 columns

---

denoised_train_df

Date	Open	High	Low	Close	Next_Day_Close
1993-06-01	451.72137393	452.63540568	450.23562740	452.00245815	452.11001990
1993-06-02	451.72137393	452.63540568	450.23562740	452.00245815	452.11001990
1993-06-03	451.72137393	452.63540568	450.23562740	452.00245815	452.11001990
1993-06-04	451.72137393	452.63540568	450.23562740	452.00245815	452.11001990
1993-06-07	451.72137393	452.63540568	450.23562740	452.00245815	452.11001990
...	...	...	...	...	...
2017-05-22	2389.18224942	2394.96833420	2383.81120983	2388.30045597	2389.94986487
2017-05-23	2389.18224942	2394.96833420	2383.81120983	2388.30045597	2389.94986487
2017-05-24	2399.09350430	2406.02942028	2397.33125988	2401.41408429	2403.03126869
2017-05-25	2399.09350430	2406.02942028	2397.33125988	2401.41408429	2403.03126869
2017-05-26	2399.09350430	2406.02942028	2397.33125988	2401.41408429	2403.03126869

6043 rows × 5 columns

Figure 4.10: Denoised Dataset Example

- **p:** The autoregressive part of the model, that allows the incorporation of the effect of past values into the model, and is the number of lag observations included in the model (lag order).
- **d:** The integrated part of the model, that includes the number of times that the raw

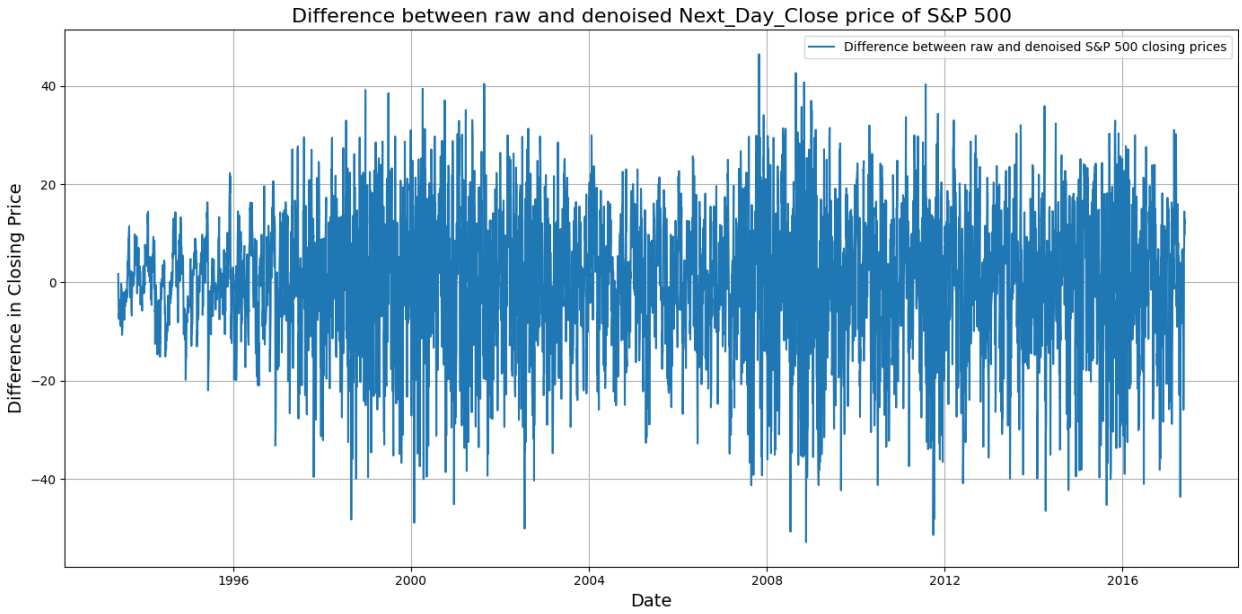

Figure 4.11: Difference Between Raw and Denoised Dataset

observations are differenced (degree of differencing).

- **q:** The moving average part of the model, that gives the size of the moving average window (order of moving average).

4.4.1.1 Optimal parameter selection with auto ARIMA

Before employing the ARIMA model, the 'auto_arima' function from the 'pmdarima' library was used to automatically determine the optimal parameters that yield the best performance for the model. The function uses a stepwise approach to search multiple combinations of p, d, and q values and selects the best model that has the lowest Akaike Information Criterion AIC, which is an estimator of prediction error and used to compare the quality of statistical models. This is a crucial step, as the performance of the ARIMA model is highly dependent on the selection of these parameters.

here was experimentation with various hyperparameters, keeping a close eye on the validation errors to guide the decision-making process. Figure 5.2 shows an example from the auto ARIMA process of the Price history with a 1-day horizon scenario, and provides the optimal parameters for the ARIMA model. The (2,1,2) are the (p, d, q) parameters of the ARIMA model.

Figure 4.13 provides the summary of auto ARIMA results for the Price history with a 1-day horizon scenario and detailed information about the model. Notably, the 'coef' column in the table gives the coefficients of the ARIMA model, and the 'P>|z|' column gives the p-values associated with these coefficients, which can be used to test the hypothesis that the coefficients are different from zero. Additionally, several tests are performed on the residuals of the model, including the Ljung-Box test for autocorrelation (independence of residuals), the Jarque-Bera test for normality, and a test for heteroskedasticity (constant variance). These tests provide further validation of the model's performance and assumptions.

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-31517.487, Time=0.55 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-31515.748, Time=0.74 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-31515.725, Time=1.10 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-31509.943, Time=0.19 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-31544.315, Time=5.41 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-31530.268, Time=2.90 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-31534.920, Time=3.18 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-31522.867, Time=2.02 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-31524.419, Time=1.33 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-31547.337, Time=3.70 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=-31546.201, Time=1.22 sec

Best model:  ARIMA(2,1,2)(0,0,0)[0] intercept
Total fit time: 22.380 seconds
```

Figure 4.12: Results of Auto ARIMA

SARIMAX Results

Dep. Variable:	y	No. Observations:	6043
Model:	SARIMAX(2, 1, 2)	Log Likelihood	15779.668
Date:	Mon, 17 Jul 2023	AIC	-31547.337
Time:	10:28:14	BIC	-31507.098
Sample:	0	HQIC	-31533.368
	- 6043		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0002	7.86e-05	2.330	0.020	2.91e-05	0.000
ar.L1	0.4499	0.139	3.228	0.001	0.177	0.723
ar.L2	0.2850	0.144	1.977	0.048	0.002	0.567
ma.L1	-0.4570	0.142	-3.226	0.001	-0.735	-0.179
ma.L2	-0.2263	0.145	-1.558	0.119	-0.511	0.058
sigma2	0.0003	1.66e-06	190.291	0.000	0.000	0.000

Ljung-Box (L1) (Q):	0.42	Jarque-Bera (JB):	127466.71
Prob(Q):	0.52	Prob(JB):	0.00
Heteroskedasticity (H):	1.83	Skew:	-0.01
Prob(H) (two-sided):	0.00	Kurtosis:	25.50

Figure 4.13: Summary Auto ARIMA Table

4.4.1.2 Auto ARIMA diagnostic plots

The auto ARIMA model also provides a set of diagnostic plots to assess the quality of the fit and the assumptions made by the model. Figure 4.14 shows the four diagnostic plots generated by the auto ARIMA model, for the Price history with a 1-day horizon scenario.

- **Standardized Residuals Over Time:** The top left plot in Figure 4.14 shows the residuals over time. In an ideal scenario, the residuals should be randomly scattered around the centerline. A specific pattern, like linear or cyclic behavior, suggests that the model can still be improved.
- **Histogram Plus Estimated Density:** The top right plot in Figure 4.14 provides a histogram of the residuals with an overlaid kernel density plot. An approximately normal distribution is a sign of a good fit.
- **Normal Q-Q Plot:** The bottom left plot in Figure 4.14 is a quantile-quantile plot of the standardized residuals. The dots should align with the red line if the residuals are normally distributed. If the dots are not aligned, then the residuals are not normally distributed, indicating that the model may not be a good fit.
- **Correlogram (ACF Plot):** The bottom right plot in Figure 4.14 is an autocorrelation plot. Ideally, for a well-specified model, the residuals should not be autocorrelated

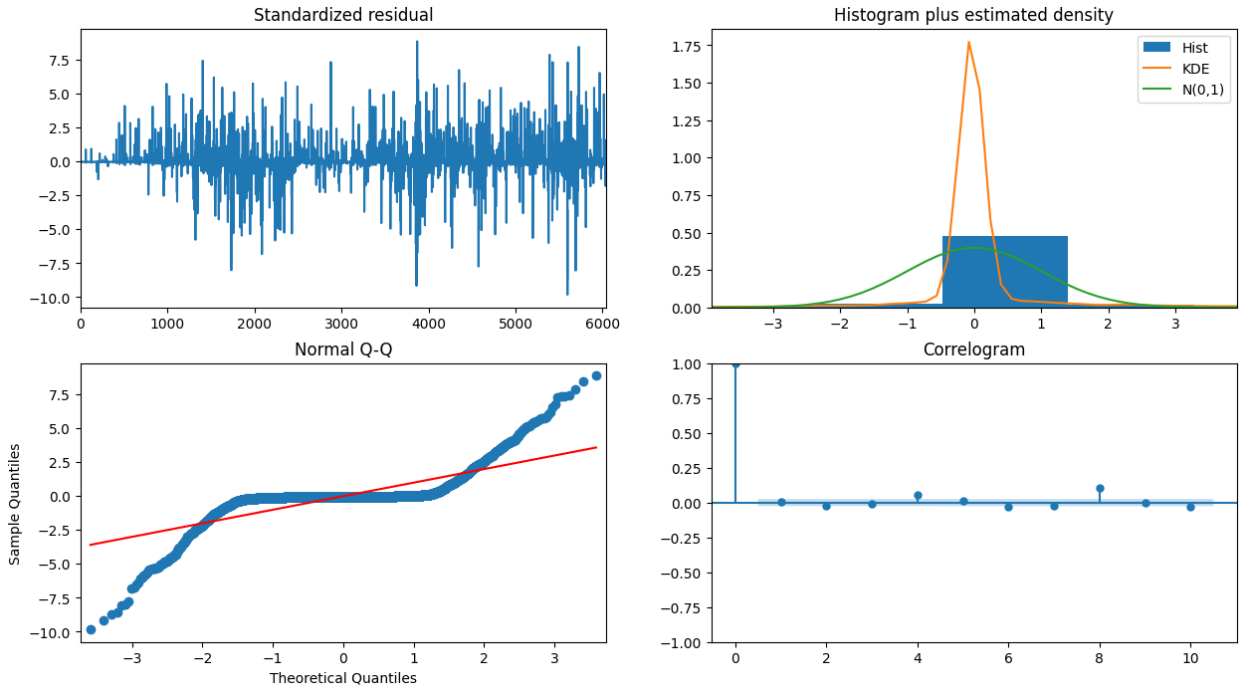

Figure 4.14: Auto ARIMA Plots

and the autocorrelation factors should be within the blue shaded region, meaning they are not statistically significant.

4.4.1.3 ARIMA model implementation

The ARIMA model implementation was carried out using the 'pmdarima' library in Python. This library offers a solid and efficient implementation of the ARIMA model, making it ideal for this use case.

This implementation covers four scenarios, each using different types of data based on forecasting horizons and sets of features:

- Price history with a 1-day horizon
- Price history with a 30-day horizon
- Multiple features with a 1-day horizon
- Multiple features with a 30-day horizon

In each scenario, the (p,d,q) parameters are gathered from the auto ARIMA results of the training set, then the model is trained on the training dataset, the performance is evaluated on the validation dataset and then tested on the test dataset.

4.4.1.4 Model performance evaluation

Validation errros: The model's performance is evaluated using several metrics including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2 Score. These metrics provide

a comprehensive view of the model's performance, and are analyzed in the previous background chapters. The Table 4.8 below summarizes the validation errors for the different scenarios.

Table 4.8: ARIMA Validation Errors for Different Scenarios

Scenario	MAE	MSE	RMSE	MAPE	R^2 Score
Price history with a 1-day horizon	29.19	1792.84	42.34	1.04	0.963
Price history with a 30-day horizon	337.74	156886.33	396.09	11.60	-2.24
Multiple features with a 1-day horizon	39.17	3528.42	59.40	1.40	0.928
Multiple features with a 30-day horizon	175.80	47861.42	218.77	6.07	0.013

Several observations can be made from these results. First, it is evident that increasing the forecast horizon from 1 day to 30 days results in larger errors. This is expected as longer-term predictions are generally more uncertain. Second, using multiple features instead of just price history appears to decrease the errors for the 30-day horizon but increase them for the 1-day horizon. This could be because the additional features contain information that is useful for longer-term predictions but add noise to the shorter-term predictions. It is worth noting the negative R^2 score for the price history with a 30-day horizon scenario. A negative R^2 can occur when the chosen model does not follow the trend of the data, so it fits worse than a horizontal line. This suggests that the ARIMA model may not be suitable for this scenario, and other models should be considered.

Actual vs predicted visualization: Additionally each scenario's actual vs predicted plots for the validation sets were plotted. Below on Figure 4.15 and 4.16, are figures of Price history with a 1-day horizon scenario, and Multiple features with a 1-day horizon scenario, indicating that ARIMA approaches the actual prices in some cases.

Residuals: Furthermore, the residuals (the differences between actual and predicted values) are examined for each scenario, to see if there's any noticeable pattern. If the model is a good fit, the residuals should appear randomly scattered around zero. A pattern in the residuals, particularly with respect to time, is an indication that the model could be improved. Residuals Figures 4.17 and 4.18 below are figures of Price history with a 1-day horizon scenario, and Multiple features with a 1-day horizon scenario, and indicate that there are no such patterns, although the models fail to capture really high downwards motion.

Through this extensive examination of the ARIMA model's performance, a deep understanding of its strengths and weaknesses in predicting the S&P 500 index can be gained.

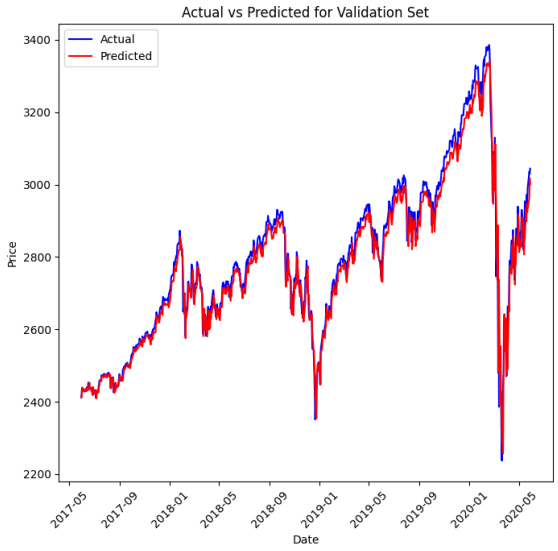

Figure 4.15: ARIMA Actual vs Predicted Price History with a 1-day Horizon

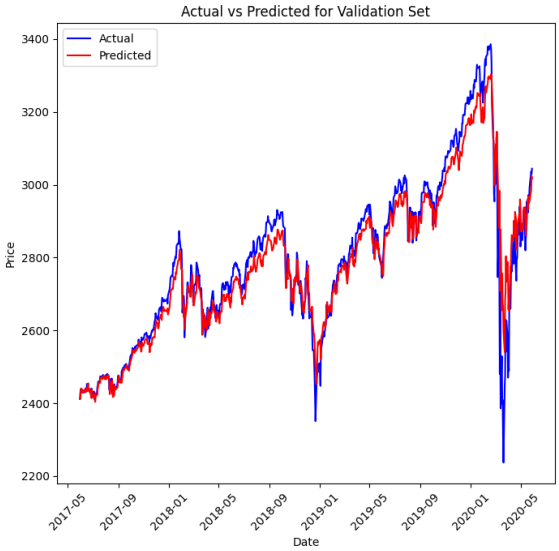

Figure 4.16: ARIMA Actual vs Predicted Multiple Features with a 1-day Horizon

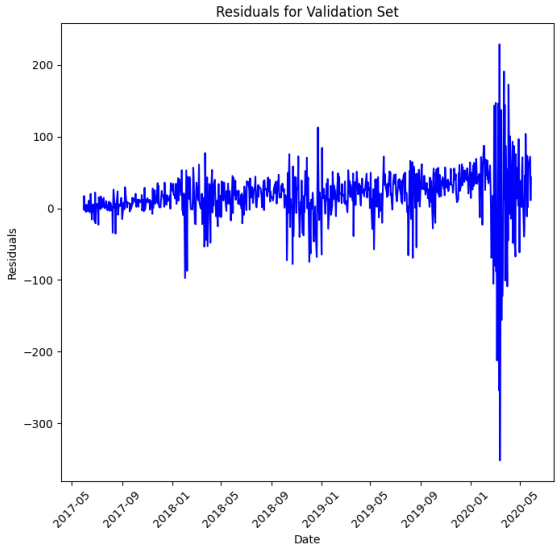

Figure 4.17: ARIMA Residuals Price History with a 1-day Horizon

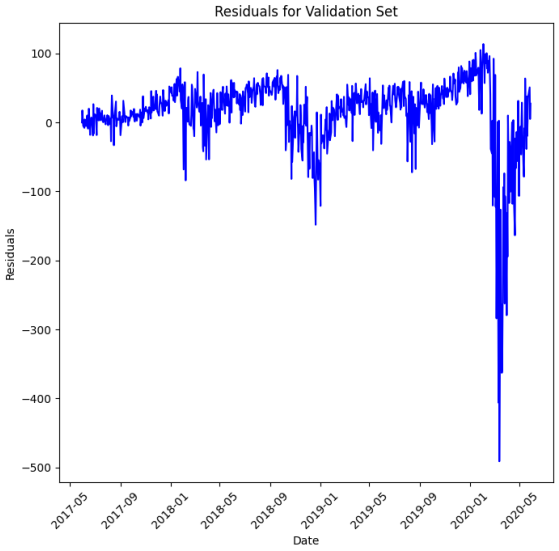

Figure 4.18: ARIMA Residuals Multiple Features with a 1-day Horizon

4.4.2 Machine learning (SVR)

The Machine learning approach implemented for the task of stock price prediction in this project utilizes Support Vector Regression (SVR), a component of the Scikit-learn library. Scikit-learn is a widely-used Python library that provides a range of supervised and unsupervised learning algorithms.

The SVR model is a version of Support Vector Machines (SVMs), an established class of machine learning algorithms primarily used for classification tasks. Unlike classical SVMs, SVR is designed to predict continuous variables.

In essence, SVR operates by mapping the input data into a high-dimensional feature space using a kernel function. Then it seeks to find a hyperplane in this space that best fits the data. The best fit is characterized by a small number of training samples, called support vectors, which lie on or nearest to the hyperplane and contribute to defining its position

and orientation. The choice of a 'linear' kernel in these scenarios was made, and implies that the separation in the high-dimensional space is linear. This linearity provides interpretability benefits, as the model's predictions are a linear function of the features.

4.4.2.1 SVR model implementation

After the preprocessing step that was performed universally, the SVR model is trained. The 'linear' kernel is used, indicating that the transformation of the data into the high-dimensional space is linear. It is important to note that the training process is conducted exclusively on the training set, thereby adhering to the fundamental practice of not using the validation and test sets in the model building process.

For comparison purposes, a Linear Regression model was also trained on the same data. However, the SVR model with the linear kernel consistently outperformed the Linear Regression model, especially for scenarios involving multiple features.

This implementation covers the four scenarios, each using different types of data based on forecasting horizons and sets of features. In each scenario, the SVR model is independently trained and evaluated:

- Price history with a 1-day horizon
- Price history with a 30-day horizon
- Multiple features with a 1-day horizon
- Multiple features with a 30-day horizon

4.4.2.2 Model performance evaluation

Validation errors: Following the training phase, the model's predictive performance is assessed. Predictions are made on the validation dataset, and prediction errors are calculated. The errors are assessed using various metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2 Score. The Table 4.9 below summarizes the validation errors for the different scenarios.

In the scenario with price history and a 1-day horizon, the model performed the best with an R^2 score of 0.973, indicating that this model could explain approximately 97.3% of the variance in the target variable. This model also had the lowest MAE, MSE, RMSE, and MAPE, suggesting that it had the smallest prediction errors among the four scenarios.

In contrast, the model trained with price history and a 30-day horizon had the highest error measures and the lowest R^2 score, suggesting that this model's predictions were less accurate. This could be due to the increased complexity and uncertainty when predicting 30 days into the future based solely on past prices.

The models trained with multiple features had improved performance over the 30-day horizon price history model, with the 1-day horizon model achieving an R^2 score of 0.956 and the 30-day horizon model achieving an R^2 score of 0.333. This demonstrates, that incorporating additional features into the models can enhance their predictive accuracy, particularly when forecasting over longer horizons.

Table 4.9: SVR Validation Errors for Different Scenarios

Scenario	MAE	MSE	RMSE	MAPE	R^2 Score
Price history with a 1-day horizon	22.07	1320.20	36.33	0.80	0.973
Price history with a 30-day horizon	120.61	37648.33	194.03	4.38	0.224
Multiple features with a 1-day horizon	30.26	2145.05	46.31	1.10	0.956
Multiple features with a 30-day horizon	124.56	32324.87	179.79	4.45	0.333

Overall, the validation errors suggest that the SVR models can provide relatively accurate predictions for stock prices, particularly when multiple features are used and when forecasting over shorter horizons.

Learning curves: A learning curve plot is an essential part of the evaluation process. The learning curve illustrates how the model's performance evolves as it is trained on an increasing amount of data of the training set size, both for the training and the validation sets. The curves plot the model's performance on both the training data and the validation data for different training set sizes. Learning curves can be very useful in diagnosing whether the model is overfitting or underfitting.

Overfitting occurs when the model performs well on the training data but poorly on the validation data. This suggests the model is too complex and is capturing the noise and outliers in the training data along with the underlying pattern. In the learning curves, overfitting is indicated by a substantial gap between the training and validation error curves.

Underfitting occurs when the model performs poorly on both the training and validation data. This suggests the model is too simple to capture the underlying structure of the data. In the learning curves, underfitting is indicated by both the training and validation error curves plateauing at a high error level.

In the context of the four scenarios examined on Figures 4.19, 4.20, 4.21, and 4.22, the generated learning curves demonstrate satisfactory performance of the SVR model. There is a convergence of the training and validation error curves as more data is used for training, suggesting that the model is not overfitting. Moreover, the errors do not plateau at a high level, which would be indicative of underfitting. The model appears to be of the appropriate complexity to capture the underlying patterns in the data across all scenarios.

Actual vs predicted visualization: For a visual comparison of the model's performance, the actual vs predicted stock prices for the validation set are plotted. This visualization enables a clear understanding of the model's predictive capabilities. Moreover, it allows for the identification of any periods where the model's predictions deviate signific-

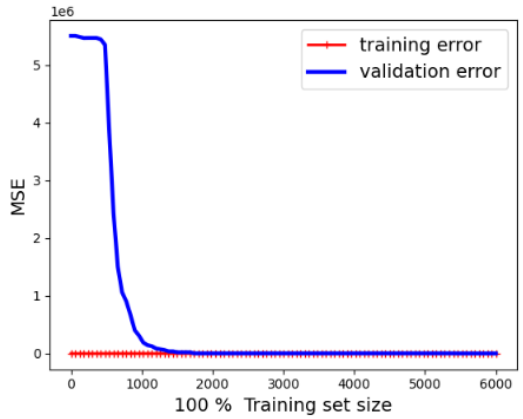

Figure 4.19: SVR Learning Curve Price History with a 1-day Horizon

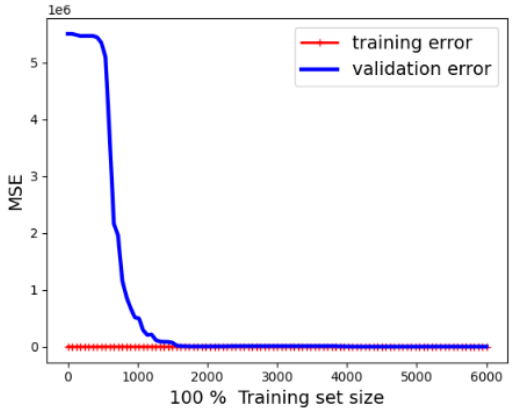

Figure 4.20: SVR Learning Curve Multiple Features with a 1-day Horizon

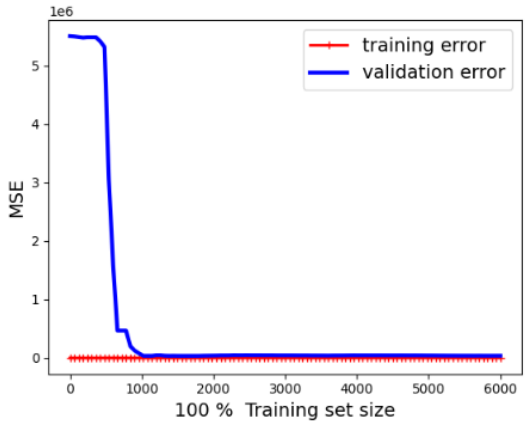

Figure 4.21: SVR Learning Curve Price History with a 30-day Horizon

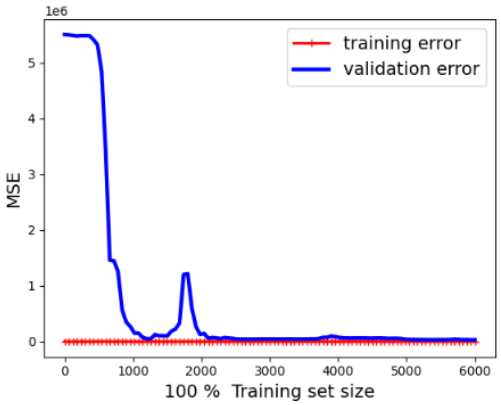

Figure 4.22: SVR Learning Curve Multiple Features with a 30-day Horizon

antly from the actual values. Below on Figures 4.23, 4.24, the plots for the 2 scenarios that predict the Next Day ahead, show exactly what the errors suggested, that models are relatively accurate, especially when multiple features.

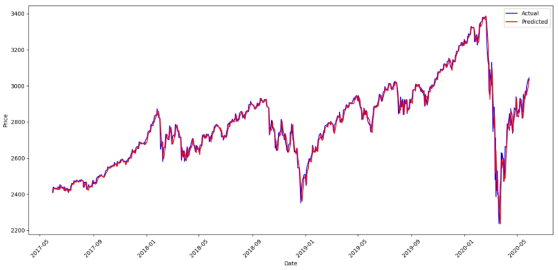

Figure 4.23: SVR Actual vs Predicted Price History with a 1-day Horizon

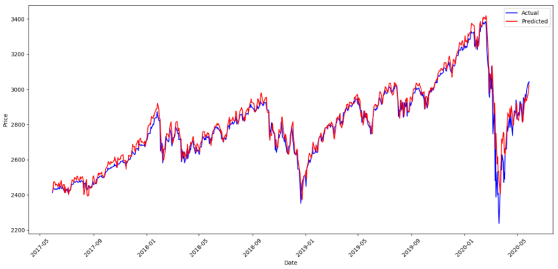

Figure 4.24: SVR Actual vs Predicted Multiple Features with a 1-day Horizon

Residuals: Finally, the residuals, which are the differences between the actual and predicted values, are plotted for the validation dataset. Analyzing the residuals can reveal any patterns that the model failed to capture from the data, which can provide insights into possible areas of model improvement. Figures 4.25, 4.26, 4.27, and 4.28, show the different residuals for the different scenarios, validating the results and the conclusions gathered from the errors.

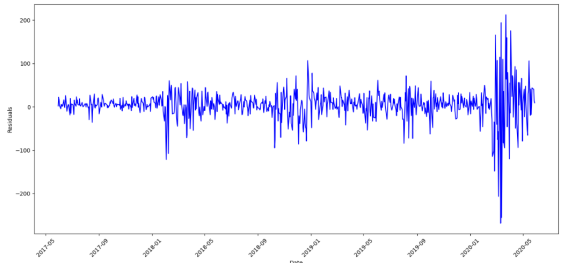

Figure 4.25: SVR Residuals Price History with a 1-day Horizon

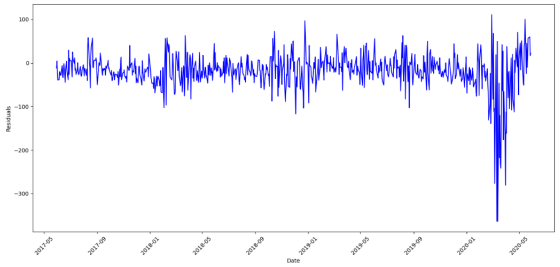

Figure 4.26: SVR Residuals Multiple Features with a 1-day Horizon

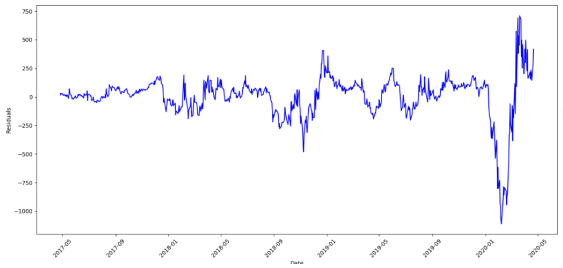

Figure 4.27: SVR Residuals Price History with a 30-day Horizon

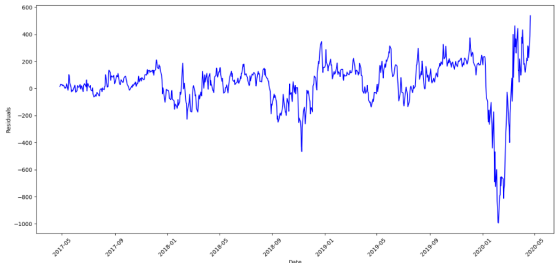

Figure 4.28: SVR Residuals Multiple Features with a 30-day Horizon

4.4.3 Deep learning (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture, that has proven to be particularly successful in modeling sequential data due to its ability to capture long-term dependencies in data. A RNN is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior.

In the implementation of LSTM in this study, the Keras library, a user-friendly neural network library written in Python, was used. Keras was developed with a focus on enabling fast experimentation. Keras runs on top of TensorFlow, an end-to-end open-source platform for machine learning.

4.4.3.1 LSTM implementation

The architecture of the implemented LSTM model consists of three layers: an LSTM layer, a Dropout layer, and a Dense layer.

The first layer is the LSTM layer, which is the main component of the model. The LSTM layer takes in a sequence of input features and processes them through the LSTM units. The number of LSTM units is a hyperparameter and refers to the dimensionality of the output space. The activation_function parameter is used to determine the activation function for these LSTM units.

The second layer is the Dropout layer, which is used to prevent overfitting during the training process. During training, some number of layer outputs are randomly ignored or ”dropped out”. The dropout_rate parameter determines the fraction of the input units to drop.

The final layer is a Dense layer, which is a neural network layer that is connected deeply, which means each neuron in the Dense layer is connected to every neuron in its previous

layer. In this case, the Dense layer has one unit and is used to output the prediction of the LSTM model.

4.4.3.2 Time window sequences

The LSTM model uses sequences to make its predictions. A sequence is a set of data where the order of the data points matters. In time series forecasting, sequences are used to provide a history of data points to the model, which it uses to make a prediction about the future.

The time window lookback sequence determines how many previous time steps to use as input variables to predict the next time period. This is specified by the time_window parameter and is also considered a hyperparameter that can be tuned.

The create_sequences function takes in a DataFrame, a target column, and a time window. It creates sequences of features (X) and the corresponding target values (y) based on the specified time window. These sequences are used to train the LSTM model.

In the context of this study, a sequence is a set of consecutive data points taken from the original time series, and the target is a data point that comes after the sequence in the time series. The model is trained to predict the target given a sequence.

It's important to note that using a time window lookback sequence transforms the shape of the dataset. For each time step in the time series, a sequence of previous time steps is provided to the model. This means that the input to the model is a three-dimensional array with dimensions [samples, time steps, features].

The creation of sequences adds an additional dimension to the data, enabling the model to learn from the temporal ordering of data points. This is particularly useful in time series forecasting, where the temporal structure of the data contains valuable information for making future predictions.

4.4.3.3 Model training

The training of the LSTM model is done using the fit function provided by Keras. The model is trained on a set of input-output pairs, denoted as X_train and y_train, respectively. The number of iterations over the entire dataset is determined by the epochs parameter, and the number of samples per gradient update is determined by the batch_size parameter.

An early stopping callback is used during training to prevent overfitting. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset. In this case, training will stop when the validation loss hasn't improved for 5 epochs, as indicated by the patience parameter. The validation loss is the value that will be monitored during training.

The validation dataset is used to provide data, that will be used to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data.

4.4.3.4 Hyperparameters

In the context of machine learning, hyperparameters are parameters whose values are set prior to the learning process. They guide the learning process and are critical to the performance of the models. For the LSTM model used in this study, several hyperparameters were considered, including the number of LSTM units, dropout rate, optimizer, learning rate, activation function, loss function, batch size, and the length of the time window lookback sequence.

The LSTM units are the dimensionality of the output space of the LSTM layer. The dropout rate is the fraction of the input units to drop, which helps to prevent overfitting. The optimizer is the algorithm used to change the attributes of the neural network such as weights and learning rate to reduce the losses. The learning rate determines the speed at which the model learns. The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it. The loss function or cost function is a method to calculate the amount of loss the model suffered while predicting the output. The batch size is the total number of training examples present in a single batch, and the time window is the number of previous time steps to use as input variables to predict the next time period.

In order to ensure the reproducibility of the experiments and maintain consistency in the comparisons across different parameters, a seed was set for all hash-based operations in Python, Numpy (used in Scikit-learn), and TensorFlow (used in Keras). All three were set with the same seed value of 42. This means that any differences in the results of the model can be attributed to the changes in hyperparameters, rather than to randomness.

Hyperparameter tuning was performed using a combination of grid search and performance evaluation metrics. Grid search is a method that performs hyperparameter tuning in a systematic way, which involves training a model for every combination of hyperparameters and retaining the best performing model.

A model was trained for each combination of these hyperparameters. The performance of each model was evaluated based on the root mean squared error (RMSE) on the validation set. The model with the lowest RMSE was considered the best model.

It's worth noting that hyperparameters were tuned separately for each of the four scenarios considered in this study:

- Price history with a 1-day horizon
- Price history with a 30-day horizon
- Multiple features with a 1-day horizon
- Multiple features with a 30-day horizon

The best set of hyperparameters was different for each scenario, reflecting the unique characteristics of each scenario's data. The results of the hyperparameter tuning for each scenario are summarized in Table 4.10.

Overall, this process of hyperparameter tuning allowed for the optimization of the LSTM model's performance on the specific tasks of this study, leading to more accurate and reliable predictions.

Table 4.10: Optimal Hyperparameters for Each Scenario

Hyper parameters	Price history (1-day horizon)	Price history (30-day horizon)	Multiple features (1-day horizon)	Multiple features (30-day horizon)
Number of LSTM units	16	4	16	64
Learning rate	0.001	0.001	0.001	0.001
Dropout rate	0.0	0.3	0.0	0.0
Optimizer	Adam	Adam	Adam	Adam
Activation function	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	MSE	MSE
Batch size	16	64	16	64
Max number of epochs (with early stop)	100	100	100	100
Time-window	5	5	5	5

4.4.3.5 Model performance evaluation

Validation errors: Evaluating the performance of a model is a critical aspect of any machine learning project. For this study, several metrics were used. These included Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The following Table 4.11 summarizes the validation errors for the four LSTM scenarios.

Table 4.11: LSTM Validation Errors for Different Scenarios

Scenario	MAE	MSE	RMSE	MAPE	R^2 Score
Price history with 1-day horizon	42.6919	4608.5012	67.8859	1.0925	0.9722
Price history with 30-day horizon	215.8195	93955.4751	306.5216	5.4113	0.4232
Multiple features with 1-day horizon	79.4616	13501.1662	116.1945	1.9700	0.9185
Multiple features with 30-day horizon	207.6510	103822.5183	322.2150	5.2252	0.3626

The LSTM model was trained and validated using different scenarios and the validation errors were calculated for each. The MAE, MSE, RMSE, and MAPE are error metrics which ideally should be minimized, while the R^2 Score (also known as the coefficient of

determination) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model, and ideally should be maximized, with the best possible score being 1.0.

From the results, it is observed that the model performs best when trained with price history with a 1-day horizon, having the lowest MAE, MSE, RMSE, and MAPE, and the highest R^2 Score. This suggests that the model is able to capture the short-term dependencies in the price history quite well. On the other hand, the model performs less optimally with a 30-day horizon and multiple features. This could be attributed to the fact that the 30-day horizon introduces a greater degree of complexity and uncertainty into the model, as it has to learn and predict over a longer period. Similarly, the addition of multiple features could increase the complexity of the model, making it harder for the model to identify and learn the important features. Additionally, even when the model is well-fitted, the validation error does not seem to reach the values of the training ones.

Learning curves: Learning curves were used to visualize the model's learning process. These curves plot the performance of the model on both the training and validation datasets over a specified number of epochs. By examining the learning curves, one can gain insight into whether the model is underfitting, overfitting, or well-fitted to the training data. Figures 4.29, 4.30, 4.31, and 4.32, that show the learning curves for the different scenarios show that the model is better fitted when predicting the next day, while the 30 days ahead doesn't perform that well. Additionally they show that a lot of noise is added with the addition of the extra features.

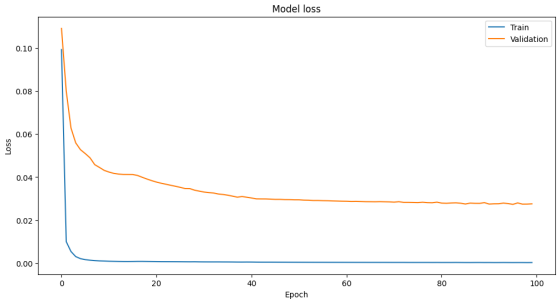

Figure 4.29: LSTM Learning Curve Price History with a 1-day Horizon

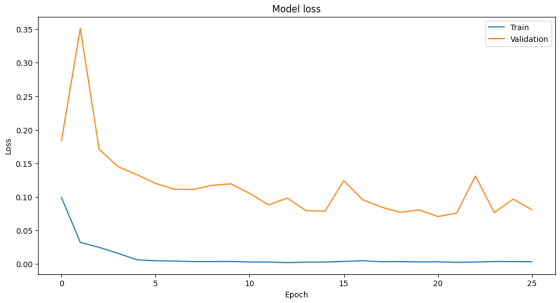

Figure 4.30: LSTM Learning Curve Multiple Features with a 1-day Horizon

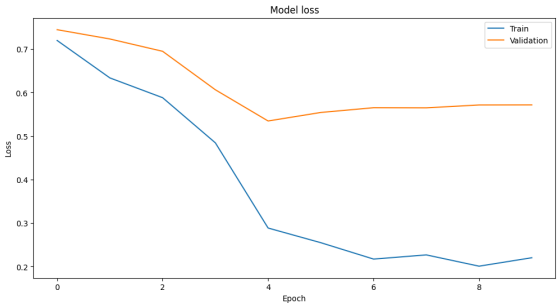

Figure 4.31: LSTM Learning Curve Price History with a 30-day Horizon

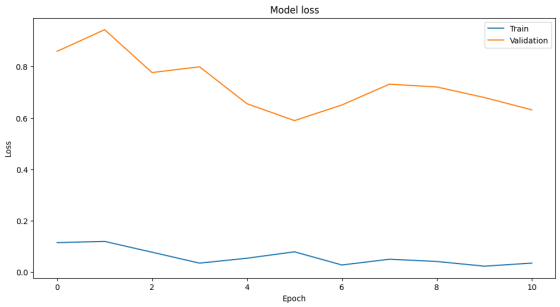

Figure 4.32: LSTM Learning Curve Multiple Features with a 30-day Horizon

Actual vs predicted visualization: A crucial part of the evaluation was the comparison of the model's predicted values against the actual values. This was visualized through

plots, which allowed for an intuitive understanding of how well the model was performing. Below on Figures 4.33, 4.34, are the plots for the 2 scenarios that predict the Next Day ahead, and show that the predictions are pretty accurate for these scenarios, with some noise for the multiple features case.

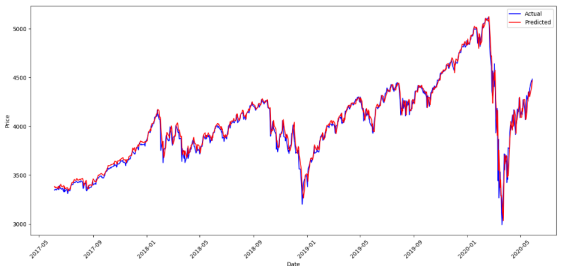

Figure 4.33: LSTM Actual vs Predicted Price History with a 1-day Horizon

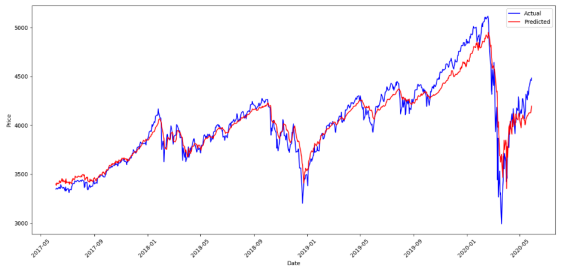

Figure 4.34: LSTM Actual vs Predicted Multiple Features with a 1-day Horizon

Residuals: Residuals, the differences between the observed and predicted values of the dependent variable, were also examined. Plots of the validation set residuals were created to ensure that they were randomly distributed around zero, indicating that the model was appropriate for the data. Figures 4.35, 4.36, 4.37, and 4.38, show the different residuals for the different scenarios, validating the results and the conclusions gathered from the errors.

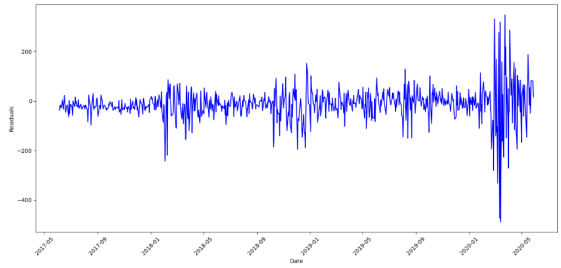

Figure 4.35: LSTM Residuals Price History with a 1-day Horizon

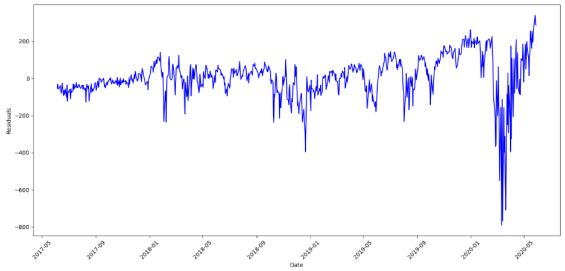

Figure 4.36: LSTM Residuals Multiple Features with a 1-day Horizon

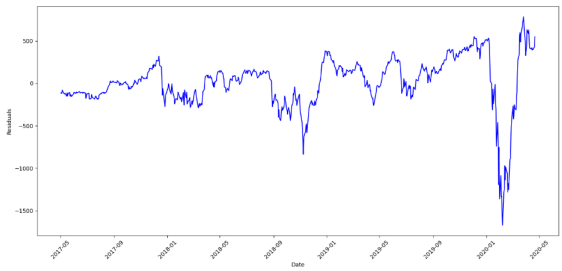

Figure 4.37: LSTM Residuals Price History with a 30-day Horizon

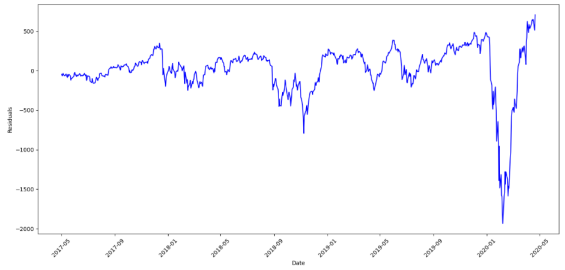

Figure 4.38: LSTM Residuals Multiple Features with a 30-day Horizon

With all these methodologies and metrics, the LSTM models were implemented, evaluated, and refined to provide reliable predictions for the given problem.

4.5 Trading simulation

The trading simulation evaluates the predictive models using different trading strategies. The primary concept is that after the market closes each day, a decision is made, whether to buy or sell stocks on the next day's opening.

4.5.1 Data used

The data used in the simulation was prepared in a way that ensures consistency and fairness. To begin with, a starting budget of $50,000 was set for all scenarios. After calculating the common date range for all the models' test dataset, the simulation ran for all common market days (723), over a date range from June 4, 2020, to April 18, 2023. This date range was selected to be common across all scenarios to allow for equal evaluation.

The data used for the simulation included predictions, real data, and dates for each model and scenario. The data was stored in a dictionary structure, with keys representing different models (ARIMA, Machine Learning, and Deep Learning) and sub-keys representing different scenarios (price history with a 1-day horizon, price history with a 30-day horizon, multiple features with a 1-day horizon, and multiple features with a 30-day horizon). For each scenario, the dictionary stores the predicted values (y_pred), the real values (y_real), and the corresponding dates (y_dates).

In addition to the prediction data, the opening and closing prices of the stocks were also extracted from the original test dataframe for each day within the simulation date range. These prices were used throughout the different trading strategies.

Although two of the scenarios used predictions with a 30-day horizon, these scenarios were handled in the same way as the other scenarios during the simulation. This decision was made to ensure equality across all scenarios. It should be noted that, in a real-world application, the strategies involving a 30-day horizon could be used differently, potentially taking more advantage of the longer-term predictions. However, for the purpose of this simulation, the decisions for these two scenarios were made based on long-term expectations, allowing us to assess their performance within the same framework as the 1-day horizon scenarios.

Having the same starting budget and date range across all scenarios, and the use of prediction, real data, and dates for each model and scenario ensures that the trading simulation is conducted on a level playing field, with each strategy having access to the same information and starting conditions. This allows for a fair comparison of the performance of each strategy.

4.5.2 Trading strategies

The core of the trading simulation lies in the various strategies employed to determine when to buy or sell stocks. Each strategy is based on a unique logic or set of rules that leverages the information available to it, including predictions from the models. It's important to note that these strategies are simulated in a simplified market environment and do not account for many complexities of real-world trading such as transaction costs, market liquidity, or the impact of trades on the market.

It is also essential to note that, in all strategies, all shares are sold on the last day of the

simulation period. This approach allows for a consistent and equitable comparison of the final budget and balance across all strategies.

4.5.2.1 Simple strategy

The first strategy is the simplest and does not consider any predictions. In this strategy, all available budget is spent to buy as many shares as possible on the first day of the simulation period at the opening price. All shares are then sold on the last day at the opening price. The strategy is purely dependent on the overall market trend during the simulation period [83].

4.5.2.2 Trend following strategy

The second strategy is based on the idea of momentum, where the direction of the stock market trend over the past day guides the investment decision for the next day [83]. If the closing price today is higher than that of yesterday, it is anticipated that the upward trend will continue, and so the strategy buys at the opening price of the next day, as many shares as the available budget allows. On the other hand, if the closing price today is lower than yesterday's, indicating a downward trend, the strategy sells all shares at the next day's opening price, expecting the downward trend to continue.

4.5.2.3 Predictive strategy

The third strategy takes into account the predictions of the models. When the model predicts that the closing price of the next day will be higher than the current day's closing price, the strategy invests all available budget in buying shares at the next day's opening price. Conversely, if the prediction is that the next day's closing price will be lower than the current day's closing price, the strategy sells all shares at the next day's opening price. This approach includes scenarios involving predictions over a 30-day horizon, where the investment decision is based on the anticipated closing price 30 days from the current day, as predicted now.

4.5.2.4 Proposed strategy

The fourth strategy, an advanced iteration of the Predictive Strategy, introduces a more measured approach to buying and selling decisions. Using the forecasted closing price for the next day, it informs the decision to either buy or sell. If the predicted closing price is higher than the current day's closing price, the strategy signals a 'buy'. Conversely, if the predicted closing price is lower, it signals a 'sell'.

However, unlike the Predictive Strategy, this strategy does not utilize the entire budget to buy shares nor does it react immediately to the 'buy' or 'sell' signal. Instead, it observes the price trend from the previous day to the current day. If a 'buy' signal was given and the current day's opening price is higher than the previous day's closing price, indicating a continued upward trend, it purchases a single share, thereby preserving the budget for future opportunities. Conversely, if a 'sell' signal was given and the current day's opening price is lower than the previous day's closing price, indicating a downward trend, it sells all shares in anticipation of further decline.

By incorporating these additional rules, this strategy aims to take advantage of both the predictive power of the model and the observed market trends, while also managing the risk associated with market volatility.

4.5.2.5 Summary of the strategies

After describing each trading strategy in detail, it is helpful to summarize the key characteristics of each in a table for quick reference and comparison. Table 4.12 below provides a high-level description of each strategy, highlighting the unique aspects of each approach.

Table 4.12: High-Level Description of Each Trading Strategy

Strategy	High-Level Description
Simple	Invests all available budget on the first day and sells all shares on the last day.
Trend-Following	Buys shares when there's an upward trend and sells when there's a downward trend.
Predictive	Uses model's prediction to make buy/sell decisions. Invests all available budget when buying shares.
Proposed	Advanced version of Predictive Strategy. Buys only one share at a time and uses model's prediction in combination with observed trends.

As shown in Table 4.12, the strategies range from a simple buy-and-hold approach to a more sophisticated strategies that use model predictions and trends. While the simple strategy is straightforward, the trend-following, predictive, and proposed strategies introduce different levels of complexity and potential for increased returns.

The results of these strategies are kept in a dictionary, which is then converted to a DataFrame for visualization. Additionally, these strategies are designed to understand how the predictions from the models can be utilized in a trading context. It's important to remember that these are simplified simulations, and real-world applications would likely involve more complex strategies and risk management techniques. Nevertheless, they provide a valuable starting point in assessing the potential usefulness of the prediction models in a trading scenario.

4.5.3 Running the Simulation

In this final part of the trading strategies evaluation, the different strategies are executed using the models' predictions and the historical stock data. Each strategy's operation yields a series of buy and sell decisions, which are then used to simulate trading activities over the selected trading period.

With an initial budget of $50,000, the trading simulations are performed for each model and scenario. This process involves applying each of the four trading strategies: 'Simple', 'Trend', 'Predictions', and 'Proposed'. The 'Simple' and 'Trend' strategies are applied directly to the historical stock data, while the 'Predictions' and 'Proposed' strategies are applied to the predictions provided by each model.

The outcome of each simulation is measured in terms of the final budget, the number of transactions executed, and the total profit or loss. The latter is calculated as the difference between the final budget and the initial budget. The results of all trading simulations are compiled into a single Pandas DataFrame, for further analysis and visualization.

By evaluating the performance of these different strategies, one can effectively assess how predictive models inform trading decisions. The results of these simulations will be presented in the next section.

5. RESULTS

This chapter presents the results of the project, summarizing the performance evaluation of each model and scenario as well as the outcome of the trading simulations. The results are analyzed in three sections. The first section focuses on the performance of each model under different scenarios. The second section discusses the outcomes of the trading simulations using the different strategies. The final section draws conclusions based on these results and provides some final thoughts.

5.1 Model and scenario performance evaluation

The first step in the results analysis is to evaluate, on the test datasets, the performance of each model (ARIMA, SVR, and LSTM) under the four different scenarios: 'Price History - 1 day horizon', 'Price History - 30 days horizon', 'Multiple Features - 1 day horizon', and 'Multiple Features - 30 days horizon'.

The evaluation is based on the metrics Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared (R2 Score). The lower the MAE, MSE, RMSE, and MAPE values, the better the model's performance. Conversely, the closer the R2 Score is to 1, the better the model's performance.

Various plots were also generated to visually compare the performance of the models. Bar plots were created for each metric, with separate plots for each model and scenario. These plots provide a graphical representation of the summary tables, making it easier to discern patterns and trends in the models' performance.

5.1.1 Summary table

The models' predictions were first aligned to a common date range to ensure fair comparison. Subsequently, the evaluation metrics were calculated for each model and scenario, and the results were compiled into a summary table. The summary Table 5.1 below, provides a comprehensive overview of the performance of each model under each scenario. Figure 5.1 shows a visual representation of the RMSE error of that table.

In the "Price History 1 day horizon" scenario, the Support Vector Regression (SVR) model performed slightly better than the LSTM, achieving the best, and lowest errors (MAE, MSE, RMSE, MAPE) and the highest R2 score.

In contrast, in the "Price History 30 days horizon" scenario, the Long Short-Term Memory (LSTM) model performed the best, despite having a negative R2 score, which indicates that the model's predictions are worse than those of a horizontal line.

For the "Multiple Features 1 day horizon" scenario, the SVR model again performed the best. However, for the "Multiple Features - 30 days horizon" scenario, the LSTM model performed the best, despite having a lower R2 score than in the 1-day horizon scenario.

These results suggest that the SVR model performs well in scenarios with a shorter prediction horizon and multiple features, while the LSTM model performs better in scenarios with a longer prediction horizon. Where the SVR is better, the LSTM is really close behind it, but when LSTM is better the SVR is far behind, making the LSTM the better choice.

Table 5.1: Summary of the Model Performance for Each Scenario

Scenario	Model	MAE	MSE	RMSE	MAPE	R2 Score
Price History 1 day horizon	ARIMA	77.984	7854.865	88.628	1.920	0.954
	SVR	36.715	2338.604	48.359	0.926	0.986
	LSTM	39.120	2627.917	51.263	0.989	0.984
Price History 30 days horizon	ARIMA	1521.977	2439094.611	1561.760	37.084	-16.763
	SVR	176.633	51120.298	226.098	4.416	0.628
	LSTM	158.047	39061.785	197.641	3.907	0.716
Multiple Features 1 day horizon	ARIMA	123.269	20353.850	142.667	2.968	0.880
	SVR	50.374	4682.677	68.430	1.256	0.972
	LSTM	83.289	10328.038	101.627	2.095	0.939
Multiple Features 30 days horizon	ARIMA	866.235	814737.227	902.628	21.037	-4.933
	SVR	370.978	177377.220	421.162	9.066	-0.292
	LSTM	187.499	52696.168	229.557	4.563	0.616

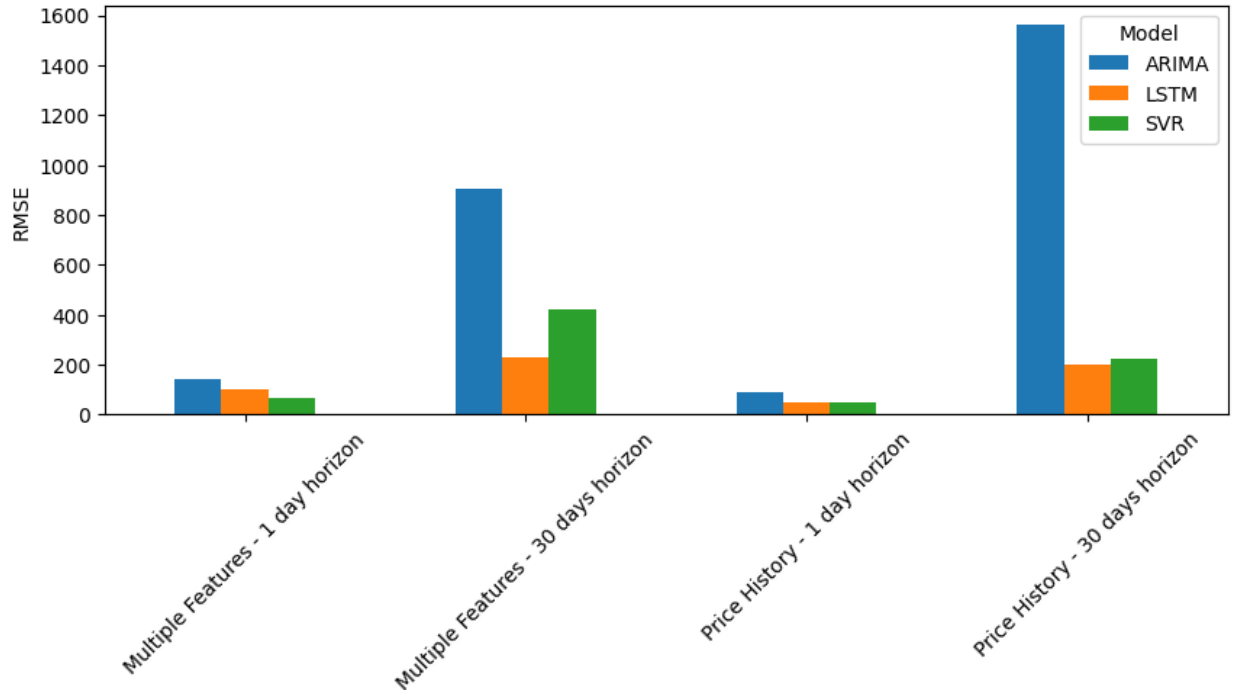

Figure 5.1: RMSE Model Performance for Each Scenario

However, it's worth noting that, the R2 scores for the 30-day horizon scenarios indicate that the predictions of all the models were poor, indicating how difficult it is to predict the long term value of the index.

5.1.2 Individual tables

The summary table can be further divided into individual tables for each model and scenario, providing a more detailed perspective of the model's performance. These tables highlight the strengths and weaknesses of each model in different scenarios, offering insights into the model's robustness and versatility.

5.1.2.1 Per model tables

ARIMA model performance metrics: Table 5.2 provides the performance metrics for the ARIMA model across all scenarios. We can observe from the table that the ARIMA model performs considerably well on the 1-day horizon scenarios but performs poorly on the 30-day horizon scenarios. Figure 5.2 shows a visual representation of the RMSE error of that table.

Table 5.2: ARIMA Model Performance Metrics

Scenario	MAE	MSE	RMSE	MAPE	R2 Score
Price History - 1 day horizon	77.984	7854.865	88.628	1.920	0.954
Price History - 30 days horizon	1521.977	2439094.611	1561.760	37.084	-16.763
Multiple Features - 1 day horizon	123.269	20353.850	142.667	2.968	0.880
Multiple Features - 30 days horizon	866.235	814737.227	902.628	21.037	-4.933

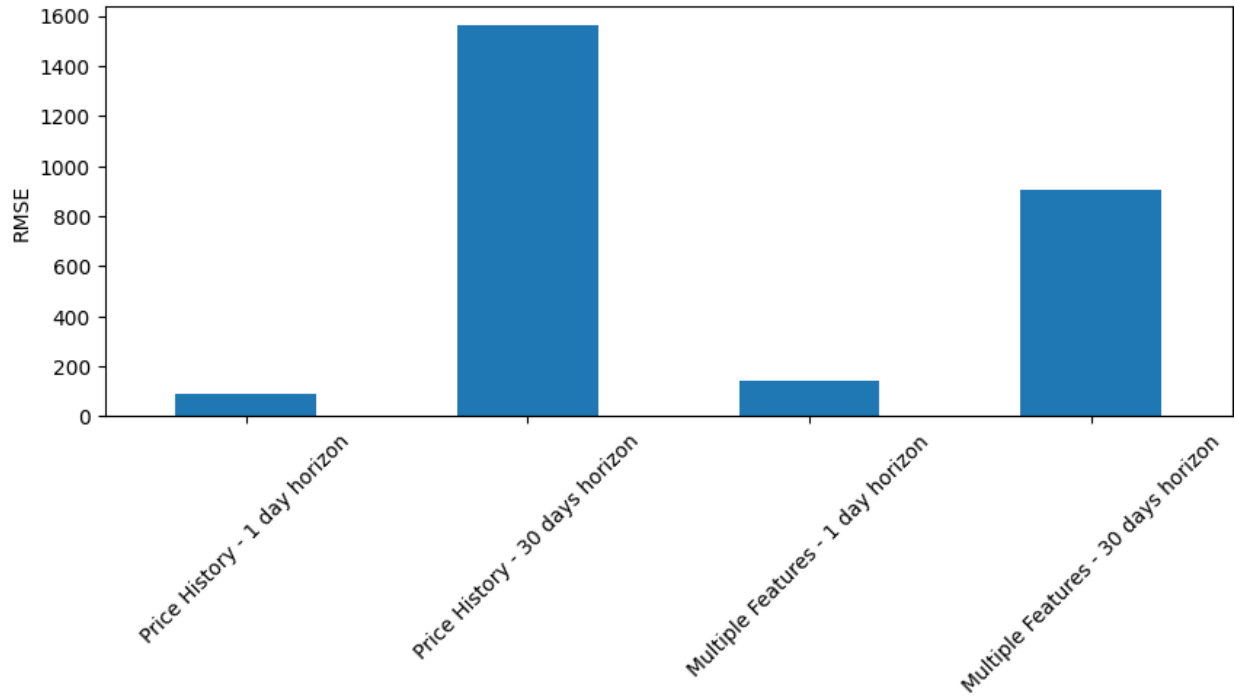

Figure 5.2: ARIMA Model RMSE

SVR model performance metrics: Table 5.3 provides the performance metrics for the SVR model across all scenarios. The SVR model shows good performance for the 1-day

horizon scenarios, but it does not perform as well on the 30-day horizon scenarios. Figure 5.3 shows a visual representation of the RMSE error of that table.

Table 5.3: SVR Model Performance Metrics

Scenario	MAE	MSE	RMSE	MAPE	R2 Score
Price History - 1 day horizon	36.715	2338.604	48.359	0.926	0.986
Price History - 30 days horizon	176.633	51120.298	226.098	4.416	0.628
Multiple Features - 1 day horizon	50.374	4682.677	68.430	1.256	0.972
Multiple Features - 30 days horizon	370.978	177377.220	421.162	9.066	-0.292

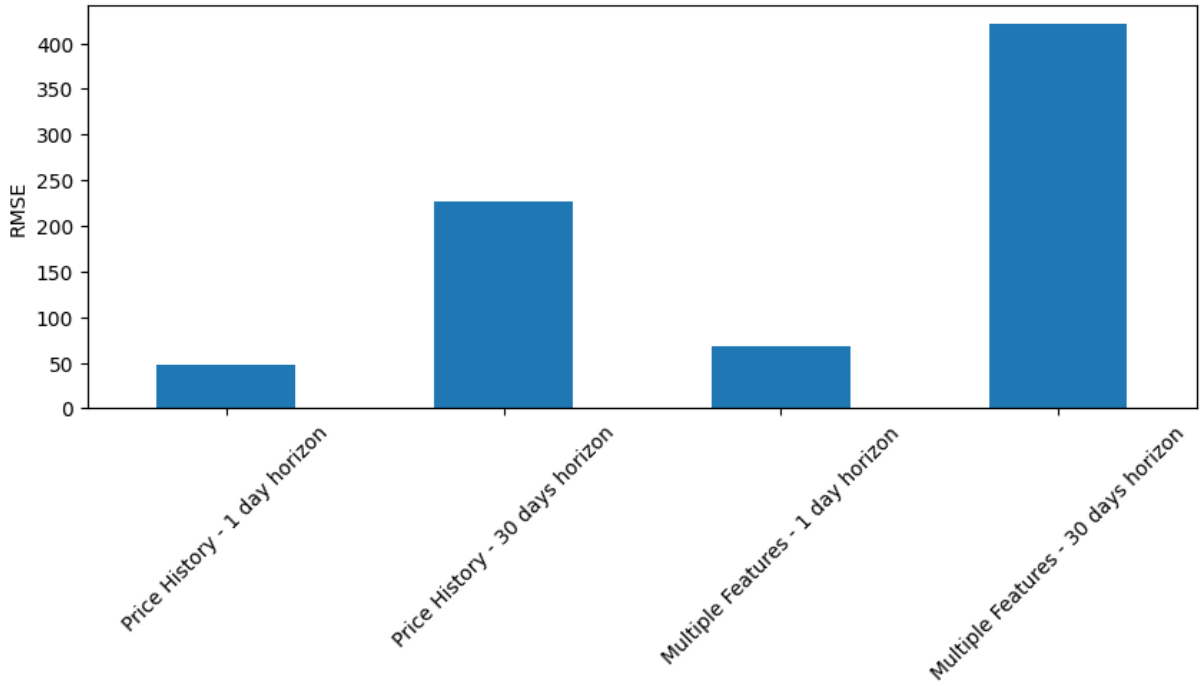

Figure 5.3: SVR Model RMSE

LSTM model performance metrics: Table 5.4 provides the performance metrics for the LSTM model across all scenarios. The LSTM model's performance is consistent across the 1-day and 30-day horizon scenarios with relatively small errors. Figure 5.4 shows a visual representation of the RMSE error of that table.

Table 5.4: LSTM Model Performance Metrics

Scenario	MAE	MSE	RMSE	MAPE	R2 Score
Price History - 1 day horizon	39.120	2627.917	51.263	0.989	0.984
Price History - 30 days horizon	158.047	39061.785	197.641	3.907	0.716
Multiple Features - 1 day horizon	83.289	10328.038	101.627	2.095	0.939
Multiple Features - 30 days horizon	187.499	52696.168	229.557	4.563	0.616

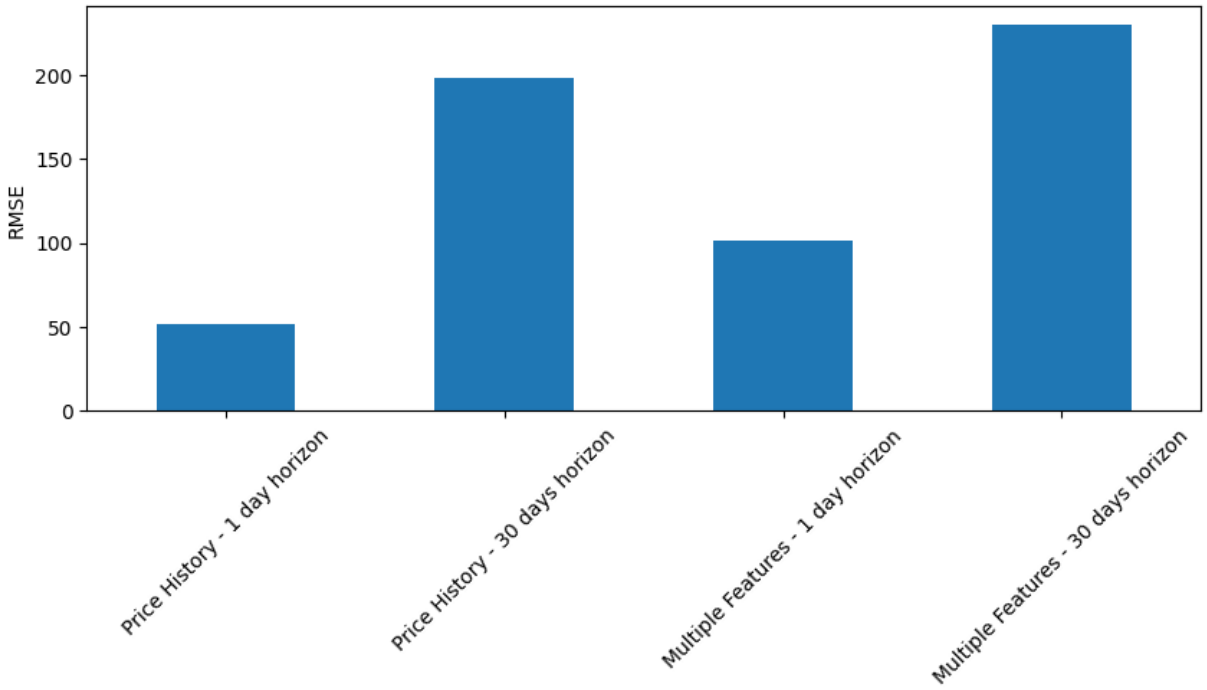

Figure 5.4: LSTM Model RMSE

5.1.2.2 Per scenario tables

Performance metrics for the 'Price History - 1 day horizon' scenario: Table 5.5 provides the performance metrics for all models in the 'Price History - 1 day horizon' scenario. The SVR and LSTM models perform similarly well in this scenario, outperforming the ARIMA model. Figure 5.5 shows a visual representation of the RMSE error of that table.

Performance metrics for the 'Price History - 30 day horizon' scenario: Table 5.6 provides the performance metrics for all models in the 'Price History - 30 days horizon' scenario. The LSTM model has the best performance in this scenario based on all metrics. Figure 5.6 shows a visual representation of the RMSE error of that table.

Table 5.5: Performance Metrics for 'Price History - 1 day Horizon' Scenario

Model	MAE	MSE	RMSE	MAPE	R2 Score
ARIMA	77.984	7854.865	88.628	1.920	0.954
SVR	36.715	2338.604	48.359	0.926	0.986
LSTM	39.120	2627.917	51.263	0.989	0.984

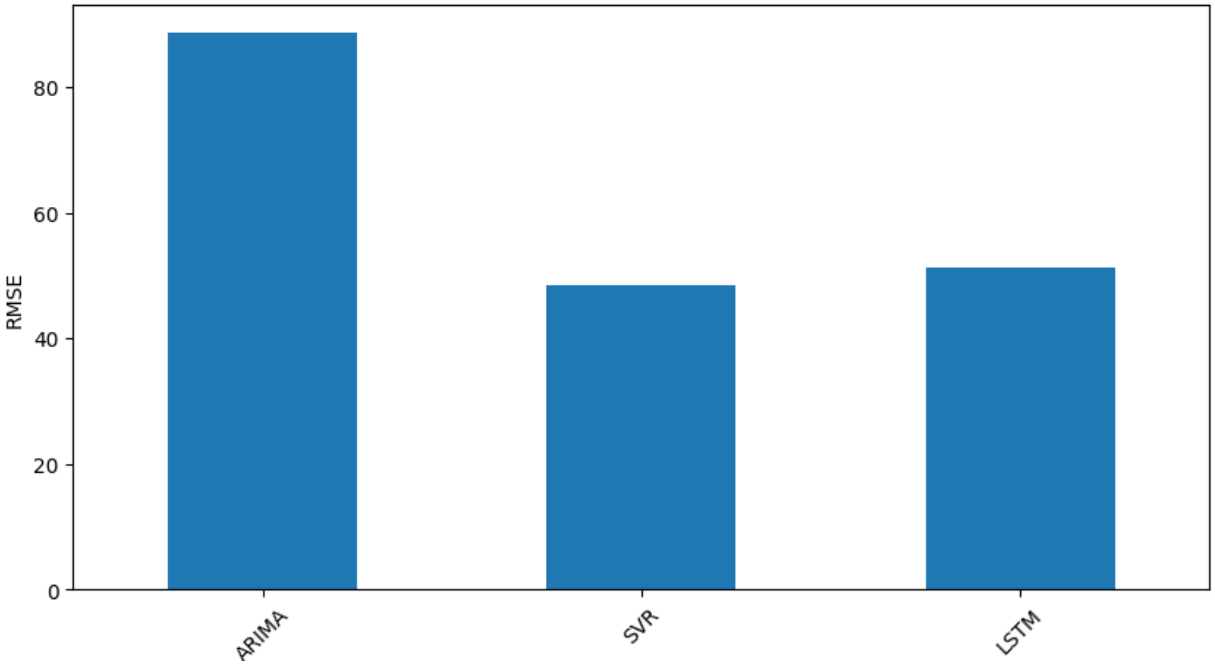

Figure 5.5: Price History - 1 day Horizon RMSE

Table 5.6: Performance Metrics for 'Price History - 30 days Horizon' Scenario

Model	MAE	MSE	RMSE	MAPE	R2 Score
ARIMA	1521.977	2439094.611	1561.760	37.084	-16.763
SVR	176.633	51120.298	226.098	4.416	0.628
LSTM	158.047	39061.785	197.641	3.907	0.716

Performance metrics for the 'Multiple Features - 1 day horizon' scenario: Table 5.7 provides the performance metrics for all models in the 'Multiple Features - 1 day horizon' scenario. All models perform well in this scenario, with the SVR model performing the best based on most metrics. Figure 5.7 shows a visual representation of the RMSE error of that table.

Table 5.7: Performance Metrics for 'Multiple Features - 1 day Horizon' Scenario

Model	MAE	MSE	RMSE	MAPE	R2 Score
ARIMA	123.269	20353.850	142.667	2.968	0.880
SVR	50.374	4682.677	68.430	1.256	0.972
LSTM	83.289	10328.038	101.627	2.095	0.939

Performance metrics for the 'Multiple Features - 30 day horizon' scenario: Table 5.8 provides the performance metrics for all models in the 'Multiple Features - 30 days horizon' scenario. In this scenario, the LSTM model outperforms the other models according to the

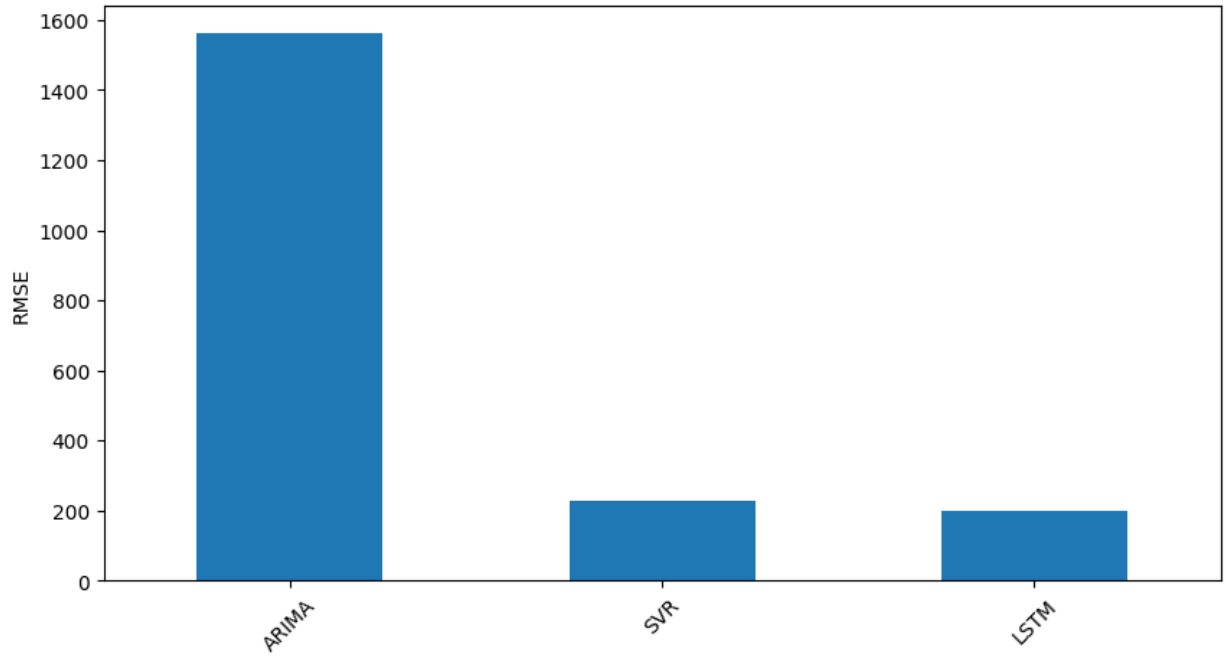

Figure 5.6: Price History - 30 day Horizon RMSE

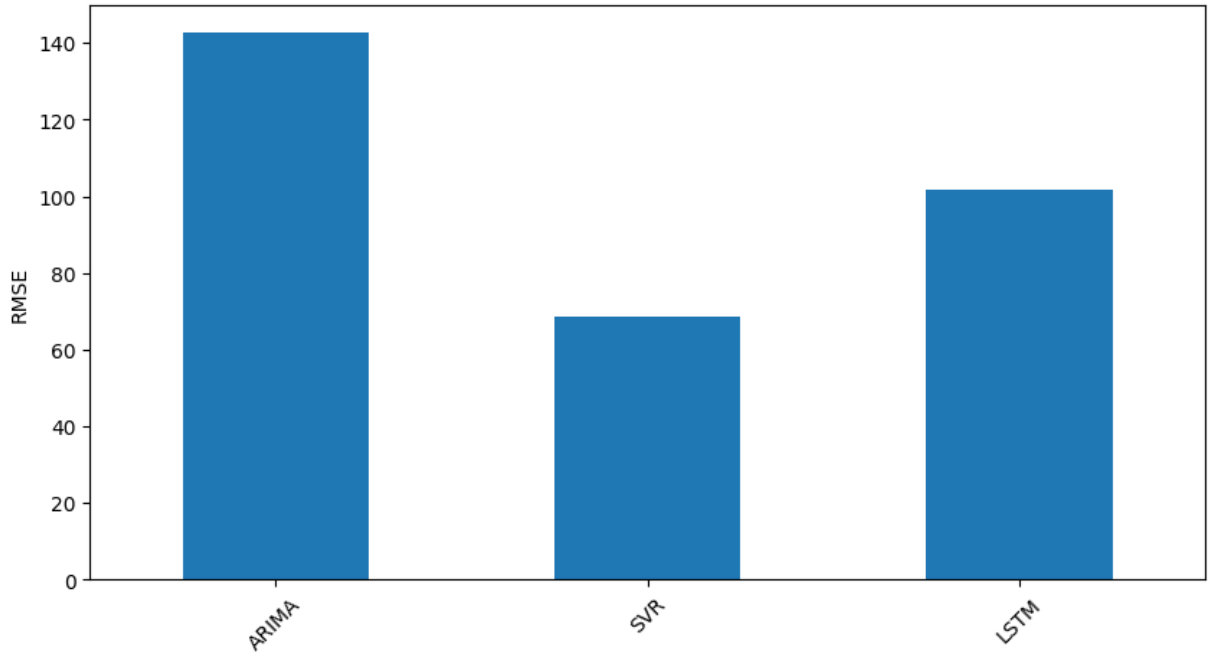

Figure 5.7: Multiple Features - 1 day Horizon RMSE

provided metrics. Figure 5.8 shows a visual representation of the RMSE error of that table.

Table 5.8: Performance Metrics for 'Multiple Features - 30 days Horizon' Scenario

Model	MAE	MSE	RMSE	MAPE	R2 Score
ARIMA	866.235	814737.227	902.628	21.037	-4.933
SVR	370.978	177377.220	421.162	9.066	-0.292
LSTM	187.499	52696.168	229.557	4.563	0.616

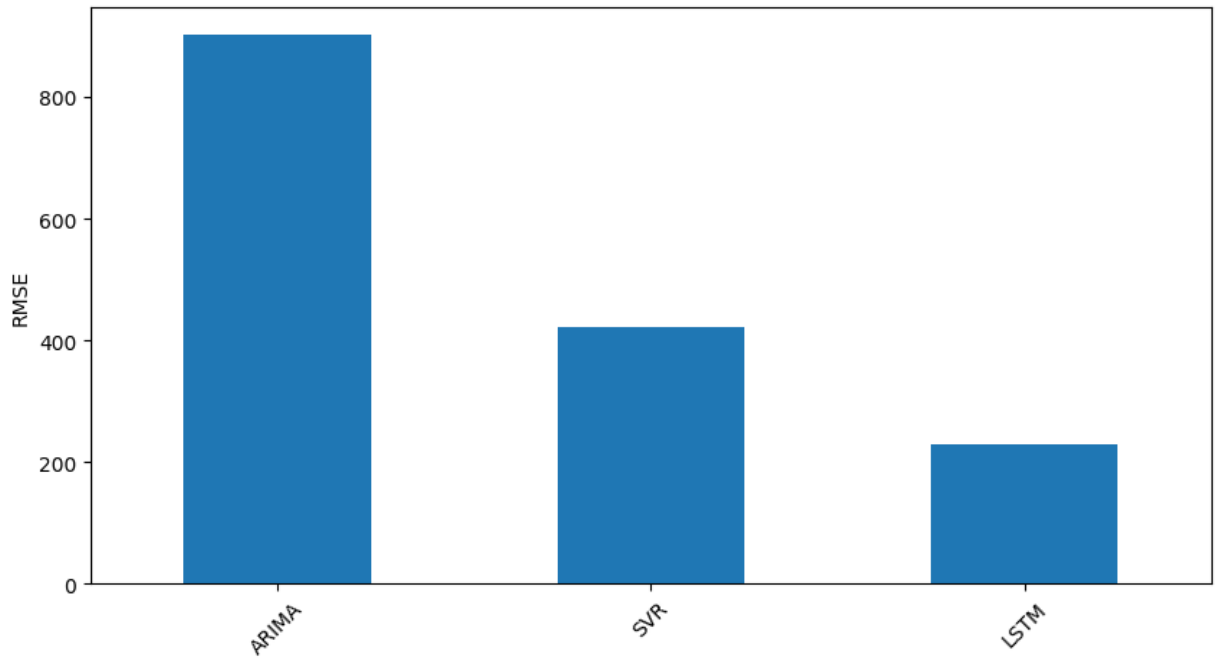

Figure 5.8: Multiple Features - 30 day Horizon RMSE

5.1.3 Grouped summary table

The evaluation metrics were also organized into a grouped summary table. This table offers another perspective on the models' performance by grouping the results based on the horizon of the prediction (1-day and 30-day) and the model used. This makes it easier to compare how well each model performed over different forecasting horizons, and understand which model excelled in which scenario.

Table 5.9 results are quite varied across the different models and scenarios. In the 1-day scenario group, the Machine Learning Model (SVR) appears to perform the best based on most metrics, with the Deep Learning Model (LSTM) also showing strong performance. In the 30-day scenario group, however, the performance of all models appears to degrade, likely due to the increased complexity and uncertainty associated with longer-term predictions. Here, the Deep Learning Model (LSTM) seems to have the best performance, particularly in the 30-day Price History scenario.

This table 5.9, provides a comprehensive summary of the performance of the three models across different scenarios, highlighting the variation in performance depending on the characteristics of the data and the prediction horizon. Figure 5.9 shows a visual representation of the RMSE error of that table.

Table 5.9: Summary of Model Performance Grouped by Scenario Group and Scenario

Model	Scenario Group	Scenario	MAE	MSE	RMSE	MAPE	R2 Score
ARIMA Model	1-day	1-day Price History	77.98	7854.86	88.63	1.92	0.95
		1-day Multiple Features	123.27	20353.85	142.67	2.97	0.88
	30-day	30-day Price History	1521.98	2439094.61	1561.76	37.08	-16.76
		30-day Multiple Features	866.23	814737.23	902.63	21.04	-4.93
Machine Learning Model	1-day	1-day Price History	36.72	2338.60	48.36	0.93	0.99
		1-day Multiple Features	50.37	4682.68	68.43	1.26	0.97
	30-day	30-day Price History	176.63	51120.30	226.10	4.42	0.63
		30-day Multiple Features	370.98	177377.22	421.16	9.07	-0.29
Deep Learning Model	1-day	1-day Price History	39.12	2627.92	51.26	0.99	0.98
		1-day Multiple Features	83.29	10328.04	101.63	2.09	0.94
	30-day	30-day Price History	158.05	39061.79	197.64	3.91	0.72
		30-day Multiple Features	187.50	52696.17	229.56	4.56	0.62

5.2 Trading simulation results

This section presents the results of the trading simulations. The trading strategies were based on predictions made by different models (ARIMA, SVR, LSTM), and their performances were evaluated based on the final balance and the total number of transactions made during the simulation period.

5.2.0.1 Table of trading results

The simulation results are grouped based on the approach, model, features, and prediction horizon used in the strategy. This allows us to compare the performance of different types of strategies and identify patterns in the results. In Table 5.10 there is a summary of the simulation results, and includes the following information:

- **Total Market Days:** the total number of days the market was open during the simulation period.
- **Total Transactions:** the total number of transactions (buying or selling) performed

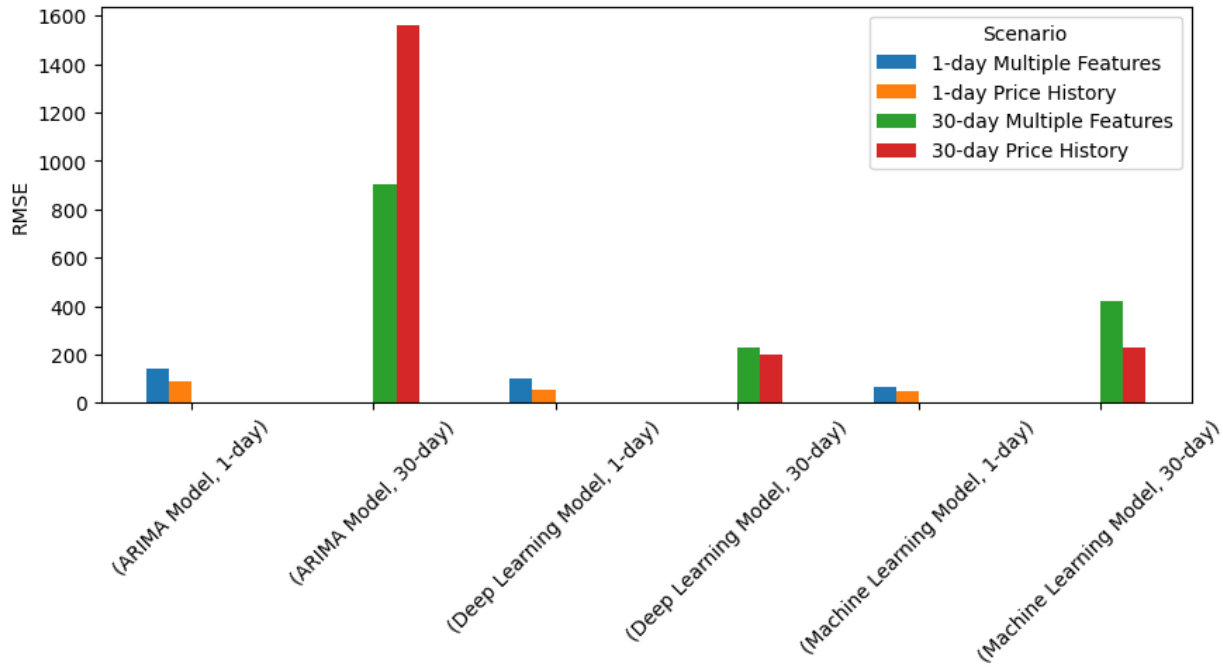

Figure 5.9: RMSE Performance Grouped by Scenario Group and Scenario

by the strategy.

- **Initial Budget ($):** the initial budget at the start of the simulation.
- **Final Budget ($):** the final budget at the end of the simulation.
- **Balance ($):** the final balance (final budget - initial budget).
- **Balance (%):** the final balance expressed as a percentage of the initial budget.

Table 5.10 that represents the trading results data.

Table 5.10: Trading Simulation Results

Approach	Model	Features	Horizon	Total Market Days	Total Transactions	Initial Budget ($)	Final Budget ($)	Balance ($)	Balance (%)
Predictions	ARIMA	Multiple Features	1	723	16	50000	54722.63	4722.63	9.45
			30	723	0	50000	50000.00	0.00	0.00
		Price History	1	723	0	50000	50000.00	0.00	0.00
			30	723	0	50000	50000.00	0.00	0.00
	LSTM	Multiple Features	1	723	79	50000	11327.24	-38672.76	-77.35
			30	723	61	50000	7320.63	-42679.37	-85.36
		Price History	1	723	170	50000	68909.54	18909.54	37.82
			30	723	65	50000	9944.99	-40055.01	-80.11
	SVR	Multiple Features	1	723	64	50000	69971.44	19971.44	39.94
			30	723	0	50000	50000.00	0.00	0.00
		Price History	1	723	354	50000	58616.14	8616.14	17.23
			30	723	262	50000	50197.92	197.92	0.40
Proposed	ARIMA	Multiple Features	1	723	15	50000	50659.87	659.87	1.32
			30	723	0	50000	50000.00	0.00	0.00
		Price History	1	723	0	50000	50000.00	0.00	0.00
			30	723	0	50000	50000.00	0.00	0.00
	LSTM	Multiple Features	1	723	179	50000	72652.65	22652.65	45.31
			30	723	154	50000	78287.61	28287.61	56.58
		Price History	1	723	246	50000	69312.52	19312.52	38.63
			30	723	139	50000	77673.27	27673.27	55.35
	SVR	Multiple Features	1	723	106	50000	64515.86	14515.86	29.03
			30	723	0	50000	50000.00	0.00	0.00
		Price History	1	723	273	50000	52577.15	2577.15	5.15
			30	723	316	50000	53026.88	3026.88	6.05
Simple	NaN	NaN	NaN	723	2	50000	66843.20	16843.20	33.69
Trend	NaN	NaN	NaN	723	240	50000	56684.94	6684.94	13.37

5.2.0.2 Visualization of trading results

A scatter plot of the balance percentage versus the total number of transactions provides another perspective on the performance of the strategies. In this plot, each point represents a different strategy, with the position along the x-axis indicating the balance percentage and the position along the y-axis indicating the total number of transactions. Figure 5.10 shows the scatter plot.

This plot allows us to visualize the trade-off between the profitability of a strategy (as measured by the balance percentage) and the activity level of the strategy (as measured by the number of transactions). Strategies that are located towards the right of the plot were able to achieve a high balance, indicating that they were able to make profitable predictions. On the other hand, strategies that are located towards the left of the plot made some wrong transactions, which may have resulted in high costs.

5.2.0.3 Breakdown of trading results

The trading simulation was performed on the S&P 500 index and the Table 5.10 presents the results of the trading simulation for different prediction strategies. Each strategy is characterized by the prediction approach (Predictions or Proposed), the model used (ARIMA, LSTM, or SVR), the type of features used (Price History or Multiple Features), and

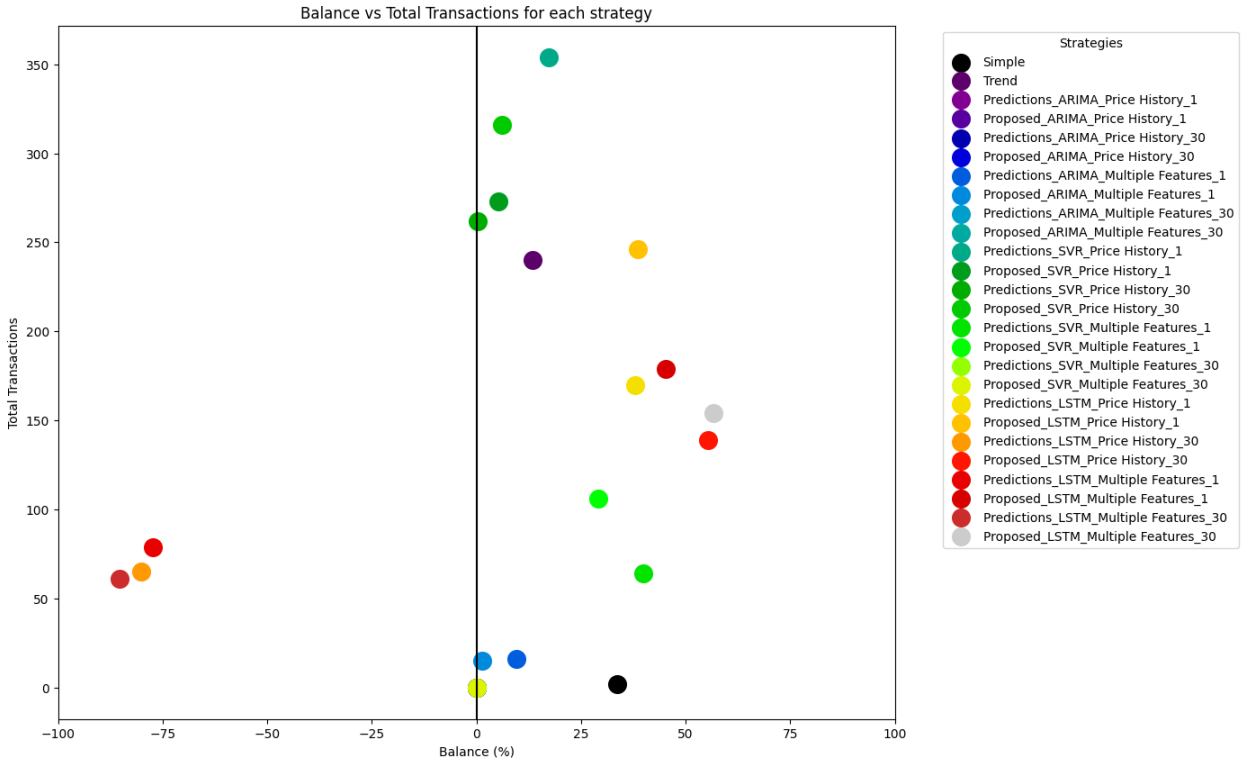

Figure 5.10: Trading Simulation Results Scatter Plot

the prediction horizon (1-day or 30-day).

The 'Balance ($)' column indicates the final balance of each trading strategy after the simulation, with the initial budget set at $50,000. A positive balance indicates a gain, while a negative balance indicates a loss. A balance of zero suggests that the final budget is equal to the initial budget, meaning that the trading strategy did not gain or lose any money.

'Simple' strategy managed to achieve a balance of 33.69%, indicating a profitable result based on the overall upward market trend during the simulation period. 'Trend' strategy achieved a balance of 13.37%, indicating its effectiveness in capturing short-term market trends. 'Predictive' and 'Proposed' had more complex results depending on each model, the features, and the horizon.

The most profitable strategies (those with the highest Balance (%)) are those using the LSTM model with the proposed approach for a 30-day horizon, both for price history and multiple features. These strategies managed to achieve a positive balance, with 56.58% and 55.35% respectively, indicating that they were able to make profitable trades based on the predictions made by the models.

The least profitable strategies are those using LSTM model with the prediction approach for a 30-day horizon, both for price history and multiple features. These strategies resulted in a negative balance, with -80.11% and -85.36% respectively, indicating that the predictions made by the models did not result in profitable trades.

The ARIMA model, regardless of the approach and features, resulted in either a small profit or no profit at all for the 30-day horizon. This suggests that the ARIMA model may not be suitable for making long-term predictions in this market.

The SVR model, on the other hand, was generally more profitable when used with the proposed approach and multiple features for a 1-day horizon, achieving a balance of 29.03%. However, it was less profitable when used with the price history for a 1-day horizon, achiev-

ing a balance of 5.15%.

In conclusion, the LSTM model with the proposed approach seems to be the most promising strategy for trading in this market, particularly for a 30-day horizon. However, the choice of features also plays a crucial role, with multiple features generally leading to more profitable trades than price history alone. The Simple and Trend approaches also show potential, despite their simplicity.

In the preceding sections, the performance of various prediction models based on metrics such as MAE, MSE, RMSE, MAPE, and R2 score was presented. These metrics measure the accuracy of the models' predictions against the actual values, which is vital for understanding the models' performance. However, when it comes to stock market trading, the real-world application of these predictions often relies more on the accurate prediction of price trends rather than the exact future value.

A model with higher error metrics can still lead to successful trading if it correctly predicts the price direction consistently. Conversely, a model with lower error metrics may not necessarily result in profitable trades if it fails to accurately predict the trend of price changes. Therefore, in this section, the focus shifted from prediction accuracy to the profitability of trading strategies based on these predictions. The resulting balance after the trading period is a more direct reflection of the practical effectiveness of these strategies in the stock market. However, it's worth noting that the strategies discussed here do not take into account transaction costs or tax implications.

6. CONCLUSION

This chapter concludes the bachelor thesis by summarizing the main findings, discussing their implications, and suggesting directions for future research.

6.1 Summary of work

This thesis embarked on the journey to explore and analyze various prediction models and trading strategies for stock market data, specifically the S&P 500 index. The objective was to assess the performance of traditional statistical methods, machine learning, and deep learning models in predicting future stock prices, and their application in formulating profitable trading strategies.

Three models were primarily focused on: ARIMA as the traditional statistical model, Support Vector Regression (SVR) as the machine learning model, and LSTM as the deep learning model. For each model, predictions were made based on two types of feature sets: price history and multiple features, and two types of prediction horizons: 1-day and 30-day. Four different trading strategies were designed and simulated, including the Simple strategy, Trend following strategy, Predictive strategy, and a Proposed strategy incorporating both predictions and observed market trends.

6.2 Main findings

The investigation into the performance of various prediction models on stock market data brought forth some intriguing findings. The prediction models' performance was evaluated using a variety of error metrics including MAE, MSE, RMSE, MAPE, and the R2 score. A key observation was that the Support Vector Regression (SVR) model performed the best (slightly better than LSTM) in scenarios with a shorter prediction horizon (1 day), whereas the Long Short-Term Memory (LSTM) model excelled in scenarios with a longer prediction horizon (30 days). This is despite the fact that the R2 scores for the 30-day horizon scenarios were less than satisfactory, underlining the difficulty of predicting long-term stock market trends.

Despite the differences in prediction accuracy, when it came to the trading simulation, the focus shifted from the exact prediction to the overall trend of the price. A model with higher error metrics could still lead to successful trading if it consistently predicted the price direction correctly.

In terms of trading strategies, the 'Simple' and 'Trend' strategies, which did not rely on any predictive models, yielded a balance of 33.69% and 13.37% respectively. This highlights the potential profitability of basic market trends and overall market direction. Among the model-based strategies, the LSTM model paired with the 'Proposed' approach proved the most profitable, particularly for a 30-day horizon. On the other hand, the least profitable strategies were those that used the LSTM model with the 'Predictive' approach for a 30-day horizon. The ARIMA model, regardless of the approach and features, led to either a small profit or no profit at all, indicating its limitations. The SVR model was in the middle profit-wise, between the other two model implementation.

6.3 Implications

These findings imply that deep learning models, specifically LSTM, coupled with a careful selection of features and a well-thought-out trading strategy, can potentially yield profitable trades in the stock market. However, the performance of these strategies is highly dependent on market volatility and trend. Therefore, while these models and strategies can guide investment decisions, they should be used in conjunction with other market indicators and personal risk assessments. Moreover the choice of model and strategy should be tailored to the specific market conditions and investment goals. It is very important, to understand market trends and incorporate them into trading strategies. Even the simplest strategies that leverage basic market trends can lead to profits. However, it is worth noting that even the most sophisticated models and strategies cannot fully eliminate the risk associated with stock market investments.

6.4 Future work

Future research can extend this work in several directions. For one, it could be interesting to experiment with other indices or even individual company stocks. This would help assess whether the findings of this thesis generalize to other stock markets or are specific to the S&P 500 index.

Further, the development and exploration of more complex models and trading strategies could provide additional insights. This includes the use of more advanced machine learning and deep learning models, the inclusion of more diverse and sophisticated features, and the use of Natural Language Processing (NLP) to incorporate news or social media sentiment into the prediction models.

Another interesting direction could be the exploration of portfolio management. Instead of focusing on a single index, future studies could explore how the models and strategies perform when applied to a portfolio of stocks. This would also provide an opportunity to investigate strategies for portfolio optimization and risk management.

Lastly, considerations like transaction costs and tax implications, which were not taken into account in this thesis, could be incorporated into the trading simulations for a more comprehensive and realistic evaluation of the strategies' performance.

Overall, the findings of this thesis provide a promising starting point for future research in the field of stock market prediction and trading strategy design.

ABBREVIATIONS - ACRONYMS

AE	Autoencoders
AI	Artificial Intelligence
ANN	Artificial Neural Networks
API	Application Programming Interface
AR	AutoRegressive
ARIMA	AutoRegressive Integrated Moving Average
CNN	Convolutional Neural Networks
CPI	Consumer Price Index
DAX	Deutscher Aktienindex
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
EMH	Efficient Market Hypothesis
FRED	Federal Reserve Economic Data
FOREX	Foreign Exchange Market
FP	False Positives
FN	False Negatives
FFNN	FeedForward Neural Network
GAN	Generative Adversarial Networks
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GDP	Gross Domestic Product
GRU	Gated Recurrent Unit
ICA	Independent Component Analysis
LSTM	Long Short-Term Memory
LR	Linear Regression
MA	Moving Average
MACD	Moving Average Convergence Divergence
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error

MLP	Multilayer Perceptron
ML	Machine Learning
MSE	Mean Squared Error
NLP	Natural Language Processing
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
RSI	Relative Strength Index
RWH	Random Walk Hypothesis
RBM	Restricted Boltzmann Machine
RMSE	Root Mean Squared Error
SGD	Stochastic Gradient Descent
S&P 500	Standard and Poor’s 500
SMA	Simple Moving Average
SVM	Support Vector Machine
SVR	Support Vector Regressor
TN	True Negatives
TP	True Positives

REFERENCES

[1] Faten Subhi Alzazah, Xiaochun Cheng, Faten Subhi Alzazah, and Xiaochun Cheng. Recent Advances in Stock Market Prediction Using Text Mining: A Survey. In *E-Business - Higher Education and Intelligence Applications*. IntechOpen, June 2020.

[2] Adebiyi A. Ariyo, Adewumi O. Adewumi, and Charles K. Ayo. Stock Price Prediction Using the ARIMA Model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, March 2014.

[3] Rob Arnott, Campbell R. Harvey, and Harry Markowitz. A Backtesting Protocol in the Era of Machine Learning. In *The Journal of Financial Data Science*, volume 1, pages 64–74, January 2019.

[4] Uma Devi B, Sundar D, and Alli P. An Effective Time Series Analysis for Stock Trend Prediction Using ARIMA Model for Nifty Midcap-50. *International Journal of Data Mining & Knowledge Management Process*, 3(1):65–78, January 2013.

[5] Belal E. Baaquie, Xin Du, Pan Tang, and Yang Cao. Pricing of range accrual swap in the quantum finance Libor Market Model. *Physica A: Statistical Mechanics and its Applications*, 401:182–200, May 2014.

[6] C. Narendra Babu and B. Eswara Reddy. A moving-average filter based hybrid ARIMA–ANN model for forecasting time series data. *Applied Soft Computing*, 23:27–38, October 2014.

[7] Yujin Baek and Ha Young Kim. ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, 113:457–480, December 2018.

[8] Malcolm Baker and Jeffrey Wurgler. Investor Sentiment in the Stock Market. *Journal of Economic Perspectives*, 21(2):129–152, June 2007.

[9] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056, November 2015.

[10] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):e0180944, July 2017.

[11] Nicholas Barberis. Investing for the Long Run when Returns Are Predictable. *The Journal of Finance*, 55(1):225–264, 2000.

[12] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, March 2011.

[13] Tung Bui. Combining Enterprise Knowledge Graph and News Sentiment Analysis for Stock Price Volatility Prediction. In *Hawaii International Conference on System Sciences*, 2019.

[14] L.J. Cao and F.E.H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, November 2003.

[15] João Carapuço, Rui Neves, and Nuno Horta. Reinforcement learning applied to Forex trading. *Applied Soft Computing*, 73:783–794, December 2018.

[16] Chen Chen, Wu Dongxing, Hou Chunyan, and Yuan Xiaojie. Exploiting Social Media for Stock Market Prediction with Factorization Machine. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 142–149, August 2014.

[17] Kai Chen, Yi Zhou, and Fangyan Dai. A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2823–2824, October 2015.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[19] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, September 2020.

[20] Sushree Das, Ranjan Kumar Behera, Mukesh kumar, and Santanu Kumar Rath. Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction. *Procedia Computer Science*, 132:956–964, January 2018.

[21] Leandro Nunes de Castro. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*. CRC Press, June 2006.

[22] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. John Wiley & Sons, February 2018.

[23] Shubharthi Dey, Yash Kumar, Snehanshu Saha, and Suryoday Basak. *Forecasting to Classification: Predicting the Direction of Stock Market Price Using Xtreme Gradient Boosting*. PESIT South Campus, October 2016.

[24] Eugene F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2):383–417, 1970.

[25] Eugene F. Fama. Random Walks in Stock Market Prices. *Financial Analysts Journal*, 51(1):75–80, January 1995.

[26] Alan Fan and Marimuthu Palaniswami. Stock selection using Support Vector Machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, volume 3, pages 1793–1798 vol.3, February 2001.

[27] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, October 2018.

[28] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, December 2018.

[29] Pushpendu Ghosh, Ariel Neufeld, and Jajati Keshari Sahoo. Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. *Finance Research Letters*, 46:102280, May 2022.

[30] Lawrence J. Gitman and Chad J. Zutter. *Principles of Managerial Finance*. The Pearson Series in Finance. Pearson, Boston, fourteenth edition, 2015.

[31] Itay Goldstein, Wei Jiang, and G Andrew Karolyi. To FinTech and Beyond. *The Review of Financial Studies*, 32(5):1647–1661, May 2019.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, November 2016.

[33] Luigi Guiso, Paola Sapienza, and Luigi Zingales. Trusting the Stock Market. *The Journal of Finance*, 63(6):2557–2600, 2008.

[34] Reza Hafezi, Jamal Shahrabi, and Esmaeil Hadavandi. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. *Applied Soft Computing*, 29:196–210, April 2015.

[35] Mark A. Hall. *Correlation-Based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.

[36] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[38] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A Survey of Forex and Stock Price Prediction Using Deep Learning. *Applied System Innovation*, 4(1):9, March 2021.

[39] Jieyun Huang, Yunjia Zhang, Jialai Zhang, and Xi Zhang. A Tensor-Based Sub-Mode Coordinate Algorithm for Stock Prediction. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 716–721, June 2018.

[40] Zulfiqar Ali Imran, Abdullah Ejaz, Cristi Spulbar, Ramona Birau, and Periyapatna Sathyanarayana Rao Nethravathi. Measuring the impact of governance quality on stock market performance in developed countries. *Economic Research-Ekonomska Istraživanja*, 33(1):3406–3426, January 2020.

[41] Weiwei Jiang. Applications of deep learning in stock market prediction: Recent progress. *Expert Systems with Applications*, 184:115537, December 2021.

[42] Neil F. Johnson, Paul Jefferies, and Pak Ming Hui. Financial markets as complex systems. In Neil F. Johnson, Paul Jefferies, and Pak Ming Hui, editors, *Financial Market Complexity*. Oxford University Press, July 2003.

[43] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2002.

[44] Vaanchitha Kalyanaraman, Sarah Kazi, Rohan Tondulkar, and Sangeeta Oswal. Sentiment Analysis on News Articles for Stocks. In *2014 8th Asia Modelling Symposium*, pages 10–15, September 2014.

[45] Ha Young Kim and Chang Hyun Won. Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications*, 103:25–37, August 2018.

[46] S. Lakshminarayanan and John P. McCrae. A Comparative Study of SVM and LSTM Deep Learning Algorithms for Stock Market Prediction. In *Irish Conference on Artificial Intelligence and Cognitive Science*, 2019.

[47] Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. On the Importance of Text Analysis for Stock Price Prediction. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1170–1175, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[48] Jiahong Li, Hui Bu, and Junjie Wu. Sentiment-aware stock market prediction: A deep learning method. In *2017 International Conference on Service Systems and Service Management*, pages 1–6, June 2017.

[49] Yuming Li, Pin Ni, and Victor Chang. An Empirical Research on the Investment Strategy of Stock Market based on Deep Reinforcement Learning model. In *4th International Conference on Complexity, Future Information Systems and Risk (COMPLEXIS 2019)*, pages 52–58, January 2019.

[50] Marcos Lopez de Prado. Beyond Econometrics: A Roadmap Towards Financial Machine Learning. *SSRN Scholarly Paper*, September 2019.

[51] Dongdong Lv, Shuhan Yuan, Meizi Li, and Yang Xiang. An Empirical Study of Machine Learning Algorithms for Stock Daily Trading Strategy. *Mathematical Problems in Engineering*, 2019:e7816154, April 2019.

[52] Pin Lv, Qinjuan Wu, Jia Xu, and Yating Shu. Stock Index Prediction Based on Time Series Decomposition and Hybrid Model. *Entropy*, 24(2):146, February 2022.

[53] Hiransha M, Gopalakrishnan E.a., Vijay Krishna Menon, and Soman K.p. NSE Stock Market Prediction Using Deep-Learning Models. *Procedia Computer Science*, 132:1351–1362, January 2018.

[54] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5.1, pages 281–298. University of California Press, January 1967.

[55] Haider Maqsood, Irfan Mehmood, Muazzam Maqsood, Muhammad Yasir, Sitara Afzal, Farhan Aadil, Mahmoud Mohamed Selim, and Khan Muhammad. A local and global event sentiment based efficient stock exchange forecasting using deep learning. *International Journal of Information Management*, 50:432–451, February 2020.

[56] Urszula Markowska-Kaczmar and Maciej Dziedzic. Discovery of technical analysis patterns. *2008 International Multiconference on Computer Science and Information Technology*, 2008.

[57] Nikola Milosevic. Equity Forecast: Predicting Long Term Stock Price Movement using Machine Learning. *Journal of Economics Library*, 3(2):288–294, June 2016.

[58] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, New York, 1997.

[59] Adil Moghar and Mhamed Hamiche. Stock Market Prediction Using LSTM Recurrent Neural Network. *Procedia Computer Science*, 170:1168–1173, January 2020.

[60] Peshawa Muhammad Ali and Rezhna Faraj. *Data Normalization and Standardization: A Technical Report*. Machine Learning Technical Reports, January 2014.

[61] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, September 2012.

[62] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, and Shahab S. Deep Learning for Stock Market Prediction. *Entropy*, 22(8):840, August 2020.

[63] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057, April 2020.

[64] Kenniy Olorunnimbe and Herna Viktor. Deep learning in the stock market—a systematic survey of practice, backtesting, and applications. *Artificial Intelligence Review*, 56(3):2057–2109, March 2023.

[65] Román Orús, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, November 2019.

[66] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, and Victor Chang. An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, 76(3):2098–2118, March 2020.

[67] Sibarama Panigrahi and H. S. Behera. A hybrid ETS–ANN model for time series forecasting. *Engineering Applications of Artificial Intelligence*, 66:49–59, November 2017.

[68] Eric Paquet and Farzan Soleymani. QuantumLeap: Hybrid quantum neural network for financial predictions. *Expert Systems with Applications*, 195:116583, June 2022.

[69] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, March 2015.

[70] Ramkrishna Patel, Vikas Choudhary, Deepika Saxena, and Ashutosh Kumar Singh. LSTM and NLP Based Forecasting Model for Stock Market Analysis. In *2021 First International Conference on Advances in Computing and Future Communication Technologies (ICACFCT)*, pages 52–57, December 2021.

[71] Nicole Powell, Simon Y. Foo, and Mark Weatherspoon. Supervised and Unsupervised Methods for Stock Trend Forecasting. In *2008 40th Southeastern Symposium on System Theory (SSST)*, pages 203–205, March 2008.

[72] David Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. *Mach. Learn. Technol.*, 2, January 2008.

[73] Stephen A. Ross, Randolph Westerfield, and Jeffrey F. Jaffe. *Corporate Finance*. The McGraw-Hill/Irwin Series in Finance, Insurance and Real Estate. McGraw-Hill/Irwin, New York, NY, 10th edition, 2013.

[74] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[75] Francesco Rundo, Francesca Trenta, Agatino Luigi di Stallo, and Sebastiano Battiato. Machine Learning for Quantitative Finance Applications: A Survey. *Applied Sciences*, 9(24):5574, January 2019.

[76] Stuart J. Russell, Peter Norvig, and Ernest Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, 3rd edition, 2010.

[77] Santosh Kumar Sahu, Anil Mokhade, and Neeraj Dhanraj Bokde. An Overview of Machine Learning, Deep Learning, and Reinforcement Learning-Based Techniques in Quantitative Finance: Recent Progress and Challenges. *Applied Sciences*, 13(3):1956, January 2023.

[78] A.J.P. Samarawickrama and T.G.I. Fernando. A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6, December 2017.

[79] G. William Schwert. Why Does Stock Market Volatility Change Over Time? *The Journal of Finance*, 44(5):1115–1153, 1989.

[80] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, May 2020.

[81] Dev Shah, Haruna Isah, and Farhana Zulkernine. Stock Market Analysis: A Review and Taxonomy of Prediction Techniques. *International Journal of Financial Studies*, 7(2):26, June 2019.

[82] Jaimin Shah, Darsh Vaidya, and Manan Shah. A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, 16:200111, November 2022.

[83] S. Shen, Haomiao Jiang, and Tongda Zhang. Stock Market Forecasting Using Machine Learning Algorithms. In *IEEE*, 2012.

[84] Hong-Gi Shin, Ilkyeun Ra, and Yong-Hoon Choi. A Deep Multimodal Reinforcement Learning System Combined with CNN and LSTM for Stock Trading. *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 7–11, October 2019.

[85] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A Comparison of ARIMA and LSTM in Forecasting Time Series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, December 2018.

[86] Hyun Sik Sim, Hae In Kim, and Jae Joon Ahn. Is Deep Learning for Image Recognition Applicable to Stock Market Prediction? *Complexity*, 2019:e4324878, February 2019.

[87] Fariha Sohil, Muhammad Umair Sohali, and Javid Shabbir. An introduction to statistical learning with applications in R: By Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013, $41.98, eISBN: 978-1-4614-7137-7. *Statistical Theory and Related Fields*, 6(1):87–87, January 2022.

[88] Haonan Sun, Wenge Rong, Jiayi Zhang, Qiubin Liang, and Zhang Xiong. Stacked Denoising Autoencoder Based Stock Market Trend Prediction via K-Nearest Neighbour Data Selection. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, Lecture Notes in Computer Science, pages 882–892, Cham, 2017. Springer International Publishing.

[89] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, Second Edition: An Introduction*. Bradford Books, Cambridge, Massachusetts, second edition, November 2018.

[90] Richard J. Teweles and Edward S. Bradley. *The Stock Market*. John Wiley & Sons, September 1998.

[91] Jonathan L. Ticknor. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 40(14):5501–5506, October 2013.

[92] Kostas Tsiveriotis. “Breaking News” and… Breaking Noise. *Journal of the Knowledge Economy*, 6(1):28–30, March 2015.

[93] Granville Tunnicliffe Wilson. Forecasting and Control,5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1. *Journal of Time Series Analysis*, 37:n/a–n/a, March 2016.

[94] Jingyang Wang, Xiaoxiao Wang, Jiazheng Li, and Haiyao Wang. A Prediction Model of CNN-TLSTM for USD/CNY Exchange Rate Prediction. *IEEE Access*, 9:73346–73354, 2021.

[95] Ju-Jie Wang, Jian-Zhou Wang, Zhe-George Zhang, and Shu-Po Guo. Stock index forecasting based on a hybrid model. *Omega*, 40(6):758–766, December 2012.

[96] Bin Weng, Mohamed A. Ahmed, and Fadel M. Megahed. Stock market one-day ahead movement prediction using disparate data sources. *Expert Systems with Applications*, 79:153–163, August 2017.

[97] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1):79–82, December 2005.

[98] Jia Wu, Chen Wang, Lidong Xiong, and Hongyong Sun. Quantitative Trading on Stock Market Based on Deep Reinforcement Learning. *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019.

[99] Maciej Wysocki and Robert Ślepaczuk. Artificial Neural Networks Performance in WIG20 Index Options Pricing. *Entropy*, 24(1):35, January 2022.

[100] Akira Yoshihara, Kazuki Fujikawa, Kazuhiro Seki, and Kuniaki Uehara. Predicting Stock Market Trends by Recurrent Deep Neural Networks. In Duc-Nghia Pham and Seong-Bae Park, editors, *PRICAI 2014: Trends in Artificial Intelligence*, volume 8862, pages 759–769, Cham, 2014. Springer International Publishing.

[101] Haiying Zhang, Qiaomei Liang, Siheng Li, Rongqi Wang, and Qingqiang Wu. Research on Stock Prediction Model Based on Deep Learning. *Journal of Physics: Conference Series*, 1549(2):022124, June 2020.

[102] Jing Zhang, Shicheng Cui, Yan Xu, Qianmu Li, and Tao Li. A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69, May 2018.

[103] Q. Zhang. Stock Trading: An Optimal Selling Rule. *SIAM Journal on Control and Optimization*, 40(1):64–87, January 2001.

[104] Xiao Zhong and David Enke. Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139, January 2017.